VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

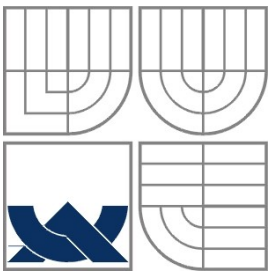
ZOBRAZENÍ ŠACHŮ POMOCÍ SLEDOVÁNÍ PAPRSKU

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

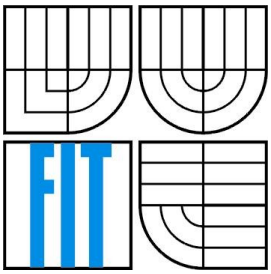
AUTOR PRÁCE
AUTHOR

BC. MARTIN VAVERKA

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ZOBRAZENÍ ŠACHŮ POMOCÍ SLEDOVÁNÍ PAPRSKU

RENDERING CHESS USING RAY TRACING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BC. MARTIN VAVERKA

VEDOUCÍ PRÁCE
SUPERVISOR

ING. ADAM HEROUT,PH.D.

BRNO 2010

Abstrakt

Tato práce se zabývá použitím metody sledování paprsku pro vykreslování 3D scény. Zachycuje výhody a nevýhody jejího použití. Popisuje úpravu této metody známou jako distribuované sledování paprsku. Dále se zabývá metodami z jiného odvětví, jako jsou konstruktivní geometrie a procedurální texturování, které s touto metodou úzce spolupracují.

Abstract

This work aims at rendering 3D scene using ray tracing. It describes advantages and disadvantages of this technology and its alternation known as distributed ray tracing. Other part deals with method from different branch, which are closely related to distributed ray tracing – constructive solid geometry and procedural texturing.

Klíčová slova

sledování paprsku, distribuované sledování paprsku, konstruktivní geometrie, procedurální texturování, osvětlovací modely, transformace objektů

Keywords

ray tracing, distributed ray tracing, constructive solid geometry, procedural texturing, shading models, tranformations of objects

Citace

Martin Vaverka: Zobrazení šachů pomocí sledování paprsku, diplomová práce, Brno, FIT VUT v Brně, 2010

Zobrazení šachů pomocí sledování paprsku

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Adama Herouta, Ph.D.Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Martin Vaverka
26.5.2010

Poděkování

Chtěl bych poděkovat Ing. Adamu Heroutovi, Ph.D. za jeho odbornou pomoc při vypracování této diplomové práce.

Obsah

1 Úvod

Metoda sledování paprsku je v počítačové grafice nejčastěji využívána pro zobrazení 3D scény. Tato metoda pracuje na principu fyzikálního šíření světla. Díky tomu má výsledný obraz vysokou fotorealističnost. Ovšem této kvality dosáhneme na úkor zvýšení výpočetní náročnosti. Proto je tato metoda používána spíše tam, kde nám zase tak o rychlost vykreslování nejde. Využívají ji nejčastěji modelovací a animovací nástroje.

Tato práce pojednává o použití metody sledování paprsku k vykreslování 3D scény. Popisuje jaké výhody a nevýhody jsou s ní spojené. Také je v ní zmíněno, jaké druhy osvětlovacích modelů lze použít pro fotorealističtěji vypadající scénu. Popíšeme si jaká je nejlepší reprezentace objektů ve spojení s touto metodou. Dále se pobavíme o netriviálnosti transformací s takovýmito objekty a o tom, jak se takové transformace řeší.

Jelikož se jedná o poměrně pomalou metodu, řekneme si něco o jejím urychlení. Dále si rozvedeme její problémy a pobavíme se o tom, jak lze tyto problémy řešit metodou distribuovaného sledování. V neposlední řadě se zmíníme o procedurálních texturách a o jejich vytváření za použití funkce Perlinova šumu.

Tato práce je postavena na základech, které řešil semestrální projekt. V něm byla řešena problematika vytváření primitiv a jejich spojování za pomoci konstruktivní geometrie do komplexnějších objektů. Semestrální projekt dále řešil použití dělení prostoru v 3D scéně, které nakonec ve výsledné práci není použito. Jediná převzatá kapitola ze semestrálního projektu je kapitola č. 4 pojednávající o konstruktivní geometrii. Ostatní problematika semestrálního projektu byla přepracována.

2 Sledování paprsku

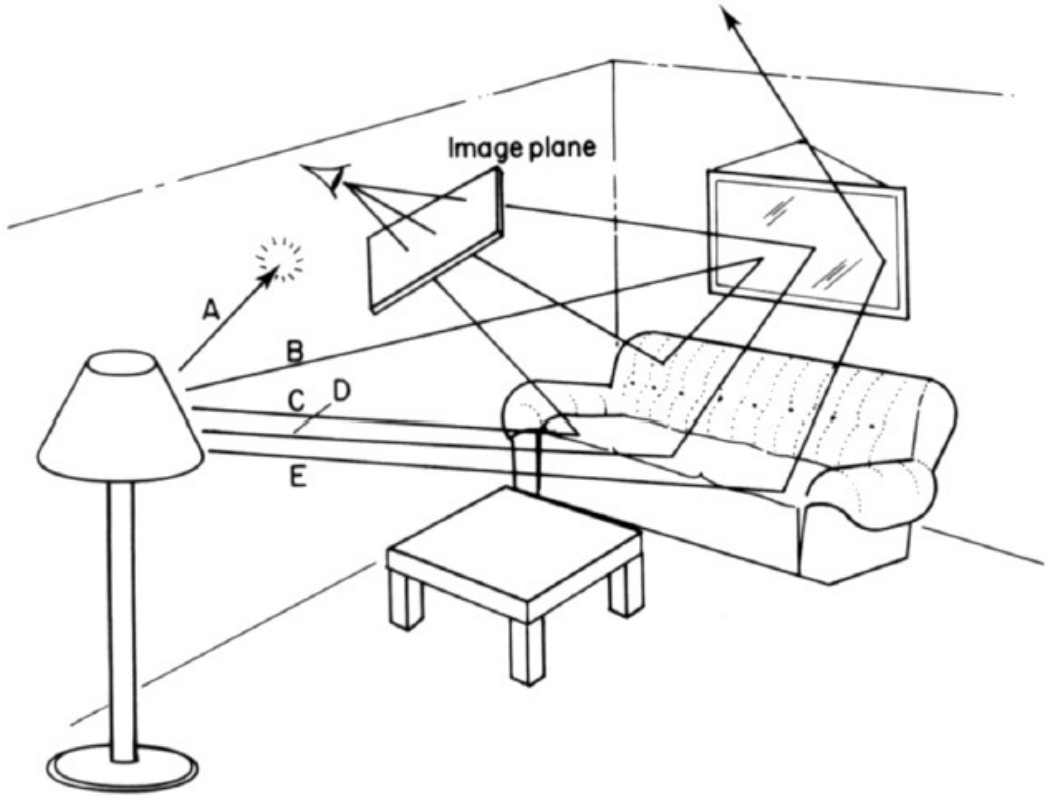
Sledování paprsku je metoda renderování 3D počítačové grafiky, přesněji se jedná o metodu globálního osvětlení. Vychází z geometrické optiky. Klasická metoda sledování paprsku má jednoduchý algoritmus. Není založena na stochastickém fyzikálním osvětlovacím modelu jako jiné metody. Při vhodné modifikaci se z ní stává základ pro komplexní algoritmy generující fotorealistickou grafiku. Nevýhodou takovýchto algoritmů je velká výpočetní náročnost. Díky tomu se tyto metody hodí spíše pro aplikace, které generují statický obraz. Hojně se využívají při tvorbě filmových a televizních speciálních efektů. Jejich použití však není vhodné pro vykreslování v reálném čase, které se používá v herních aplikacích.

2.1 Princip metody

V reálném prostředí vycházejí paprsky světla ze světelného zdroje a při interakci s objektem se odráží a lámou. My jsme schopni vidět ty objekty, od nichž se paprsky odrážejí do našeho oka. Na tomto principu je metoda sledování paprsku založena. Existují dva přístupy jak vypočíst šíření světla pomocí geometrické optiky. Můžeme sledovat šíření fotonů nebo sledovat paprsek směrem od pozorovatele.

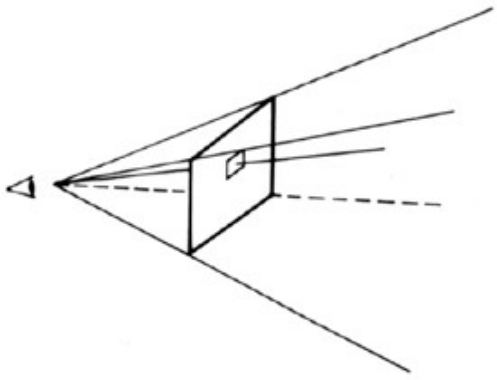
Zvolíme-li přístup sledování fotonů, vysíláme paprsky ze světelného zdroje ve směru šíření světla a sledujeme, na jaká tělesa dopadají, kam se odráží, lámou a jak je objekt absorbuje. Pokud foton při své cestě nakonec průmětnu protne, poznamenáme si jeho barvu v příslušném pixelu. Tato varianta má jednu obrovskou nevýhodu. Většina fotonů skrze průmětnu vůbec neprojdou a tedy jejich sledování bylo zbytečné. Sledování fotonů se proto v klasické metodě sledování paprsků moc často nevyužívá.

Tohoto principu se ale využívá při „mapování fotonů“ (angl. photon mapping). Jedná se o metodu globálního osvětlení, při které jsou paprsky vysílány jak ze zdroje světla tak i od pozorovatele. Oba směry jsou sledovány nezávisle a při splnění určitého kritéria jsou spojeny a je vypočítána hodnota záření. Tato metoda se využívá pro realistickou simulaci interakce světla s různými objekty. Konkrétně se využívá k simulaci lomu světla u průhledných látek jako jsou sklo nebo voda [3].



Obr.2.1. Znázornění šíření světla pomocí paprsků
Převzato z [4]

Chceme-li sledovat paprsek směrem od pozorovatele, vrháme paprsky od něj skrze projekční plochu do scény. Přeformulujeme si problém „nalezení fotonu, který trefí průmětnu“ na „jakou barvu má paprsek přilétající k pozorovateli skrze bod projekční plochy“. Ve skutečnosti barvu paprsku ovlivňuje několik fotonů. Našim úkolem je zjistit, o které fotony se jedná a jak přispívají k výsledné barvě paprsku. Tento princip využívá většina algoritmů založených na metodě sledování paprsků [4].



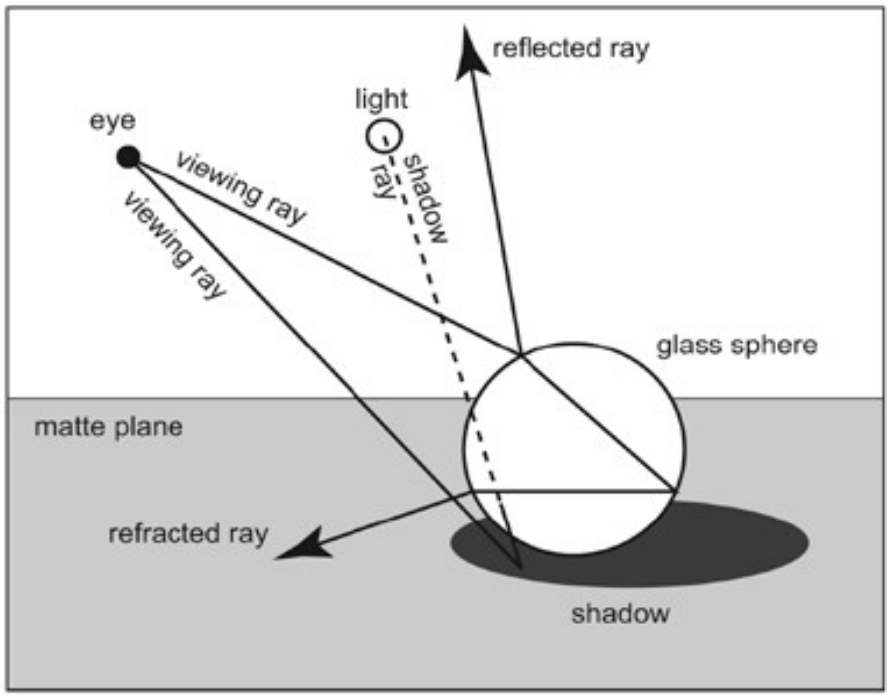
Obr. 2.2. Vyslání paprsku pixelem v průmětně
Převzato z [4]

2.2 Vypočtení barvy paprsku

Zajímáme se o to, jak vznikl paprsek, který přichází k pozorovateli daným pixelem průmětny. Snažíme se tedy nalézt nejbližší povrch, který paprsek protíná. Od tohoto povrchu totiž paprsek přichází. V závislosti na vlastnostech povrchu světlo tvoří následující složky: ambientní osvětlení, světlo vzniklé přímým difúzním odrazem světelných zdrojů od povrchu, světlo vzniklé přímým zrcadlovým odrazem světelných zdrojů, zrcadlový odraz paprsku přicházejícího z jiného objektu v odpovídajícím směru, světlo lomené na povrchu objektu [4]. Všechny tyto složky bereme v úvahu při zjištění barvy, kterou paprsek nese. Jakou má paprsek výslednou barvu vypočteme pomocí osvětlovacího modelu.

2.3 Typy paprsků

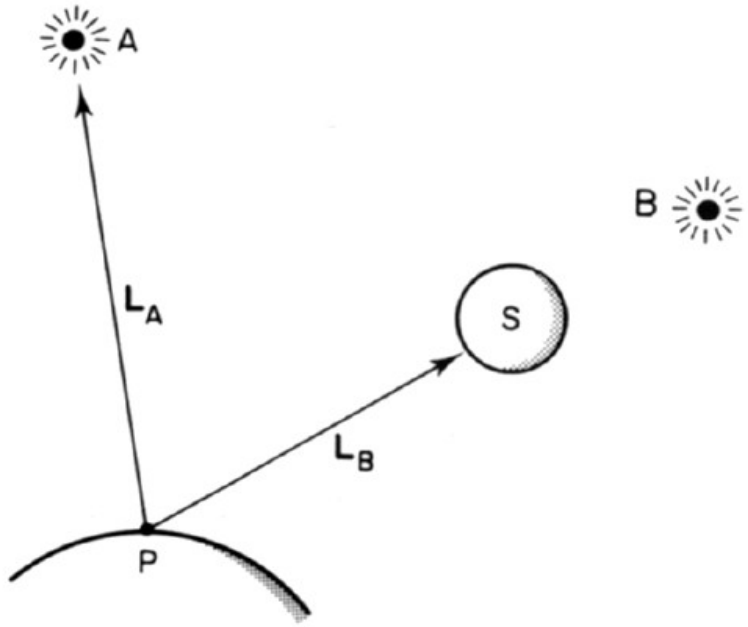
Dříve, než se začneme bavit o osvětlovacím modelu, tak si vysvětlíme, jaká terminologie je spojena s jednotlivými paprsky. Máme jeden primární paprsek (angl. viewing ray) a několik sekundárních paprsků, mezi které patří stínový paprsek (angl. shadow ray), odražený paprsek (angl. reflected ray) a lomený paprsek (anlg. refracted ray).



Obr. 2.3. Typy paprsků: Viewing ray – primární paprsek, shadow ray – stínový paprsek, reflected ray – odražený paprsek, refracted ray – lomený paprsek. Převzato z [4]

Primární paprsek je paprsek, o jehož barvu se zajímáme. Je to paprsek, který vrháme od pozorovatele skrze bod průmětny do scény. Z místa dopadu primárního paprsku jsou následně vyslány sekundární paprsky.

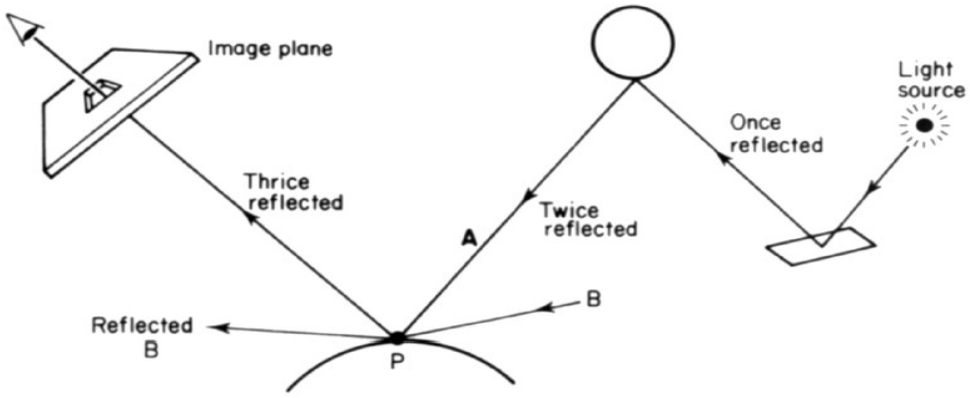
Stínové paprsky jsou zodpovědné za zobrazení stínů. Jsou posílány směrem ke zdrojům světa a testují, zda existuje nějaká překážka, která by paprsku světla zatarasila cestu.



Obr. 2.4. Stínové paprsky. Převzato z [4]

Odražené paprsky jsou posílány směrem, z kterého by muselo přijít světlo, aby se dostalo k pozorovateli. Jak uvádí [4], odraz paprsku **v** od povrchu s normálou **n** vypočteme jako:

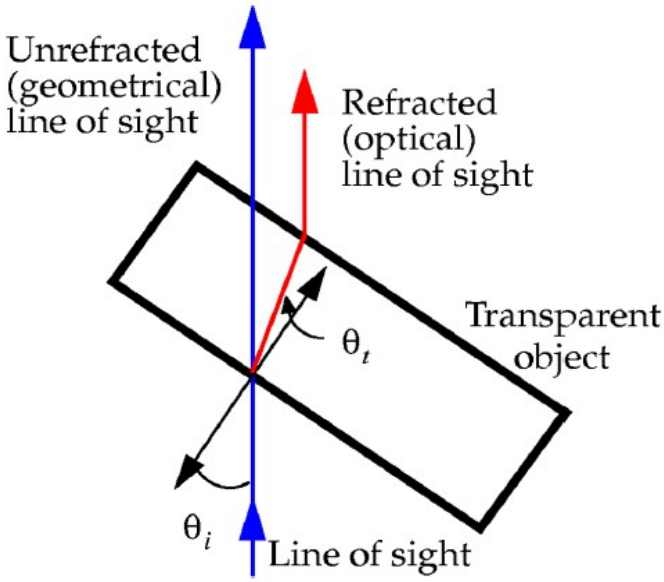
$$\mathbf{r} = 2(\mathbf{n} \cdot \mathbf{v})\mathbf{n} - \mathbf{v} \tag{2.1}$$



Obr. 2.5. Odražené paprsky. Převzato z [4]

Lomené paprsky jsou pak, podobně jako odražené paprsky, vyslány směrem, odkud by muselo projít světlo povrchem, aby proťalo projekční rovinu a dostalo se k pozorovateli. Směr lomeného paprsku vypočteme pomocí Snellova zákonu lomu, který je založen na principu nejkratšího času průchodu paprsku prostředím.

$$\frac{\sin\theta_i}{\sin\theta_t} = \frac{n_t}{n_i} \tag{2.2}$$



Obr.2.6. Lomené paprsky. Převzato z [4]

2.4 Osvětlovací model

Tato podkapitola byla převzata z [5]. V reálném světě je interakce světla s povrchem značně komplikovanou záležitostí. Existují zdroje světla, které vyzařují elektromagnetické vlnění o určité délce. Na toto vlnění reagují body na povrchu předmětů a podle vlastností povrchu i oni sami vyzařují světlo. Předměty se tak stávají sekundárními zdroji světla. Interferencí vln vyzařovaných z předmětu pak vznikají různé efekty jako je matný nebo lesklý povrch, průhlednost předmětů a jiné efekty. Kdybychom se snažili vytvořit takovýto světelný model v počítačem simulovaném prostředí, tak bychom strávili spoustu času nad výpočty s ním spojenými.

V počítačové grafice užíváme značně zjednodušené modely na výpočet osvětlení. Při jeho výpočtu se přikláníme k Newtonovu geometrickému modelu šíření světla, tedy bereme paprsky světla jako přímočaré. Opuštěním vlnového modelu si značně zjednodušíme práci. Touto úpravou se dostáváme do vektorové matematiky, kde se nám snadno šíření světla popisuje. Opuštění vlnového

modelu má ale i své nevýhody. Těžko se nám popisují některé jevy, jako jsou polarizace, ohyb světla nebo interference.

Bereme-li v úvahu, že světlo dopadající ze zdroje světla se na povrchu odráží a osvětluje jiný povrch, kde se opět odráží, tak můžeme říci, že ve scéně máme značné množství světla, u kterého těžko můžeme určit, z jakého zdroje původně pochází. Takovému světlu říkáme světlo ambientní. Uvažujeme, že ambientní světlo se ve scéně nachází zhruba ve stejném množství. Jak je toto světlo od povrchu odráženo záleží na vlastnostech materiálu. Jinak toto světlo odráží matné objekty a jinak lesklé. Ovšem nelze říci, že bychom měli jenom matné či lesklé objekty. Právě naopak většina objektů má vlastnost něco mezi.

2.4.1 Phongův osvětlovací model

Tato kapitola vytvořena na základě [5] a [6]. Jak jsou tyto jednotlivé složky spojeny určuje osvětlovací model. Nejpoužívanějším modelem je v počítačové grafice bezpochyby Phongův osvětlovací model, jehož autorem je Bui Tuong Phong. Podle něj se výsledné světlo vypočítává podle následujícího vzorce:

$$I_p = k_a i_a + \sum_{m \in \text{světla}} (k_d (L_m \cdot N)\, i_d + k_s (R_m \cdot V)^{\alpha} i_s) \tag{2.3}$$

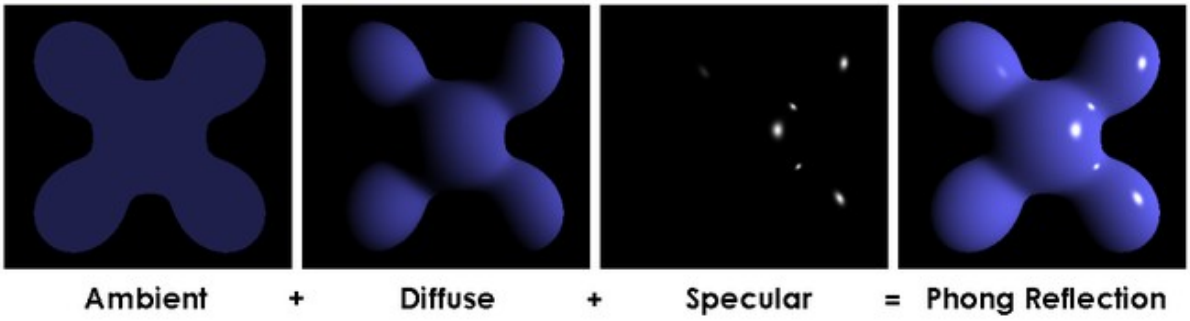
a - ambientní složka, d - difúzní složka, s - spekulární složka, k - konstanta, i - intenzita, L - směr ke zdroji světla, N - normálový vektor, R - směr odrazu světla, V - směr k pozorovateli, α - zářivost materiálu

Chceme-li do výpočtu zahrnout i odražené nebo lámané paprsky, vzorec bude mít podobu:

$$I_p = k_a i_a + \sum_{m \in \text{světla}} (k_d (L_m \cdot N)\, i_d + k_s (R_m \cdot V)^{\alpha} i_s) + k_r S_r + k_t S_t \tag{2.4}$$

S - sekundární paprsek, r - odrazová složka, t - lomená složka

Z uvedených vzorců je vidět, že Phongův model je hrubou aproximací. Lepší výsledky zajistíme nahrazením výpočtů dílčích složek sofistikovanějšími metodami. Můžeme například nahradit výpočet difúzní složky Oren-Nayarovým modelem, lesklé složky pak Blinnovým nebo Torrance-Sparrowovým modelem.

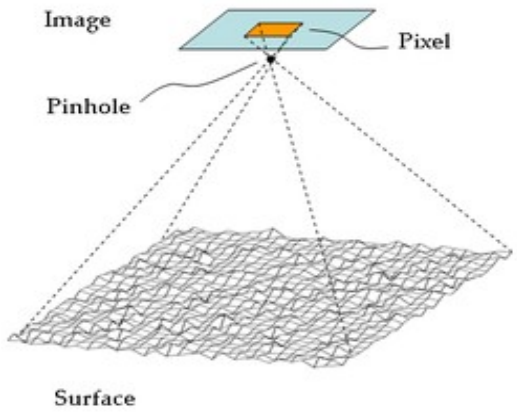


Obr. 2.7. Phongův osvětlovací model. Převzato z [6]

2.4.2 Oren-Nayarův model odrazivosti

Tato kapitola byla převzata z [7]. Oren-Nayarův model odrazivosti, vyvinutý Michaelem Orenem a Shree K. Nayarem, je model odrazivosti pro difúzní odraz od drsných povrchů. Ve většině případů se v počítačové grafice difúzní složka vypočítává podle Lambertova zákona. Ten říká, že intensita záření, kterou je možno pozorovat na tzv. „Lambertově" povrchu, je přímo úměrná kosinu úhlu mezi přímkou od pozorovatele a kolmicí k povrchu. Povrchy, které mají takto vypočítanou difúzní složku se jeví ze všech směrů stejně jasně.

Oren-Nayarův model uvažuje povrch jako množinu faset (hran skosených pod daným úhlem) s různými úrovněmi skosení. Vzhledem k tomu, že foto receptory sítnice mohou detekovat jen konečnou oblast, tak se významné makroskopické části drsného povrchu podílejí na výpočtu celkové jasové hodnoty ve větší míře než ostatní. Tím je zaručeno, že pod různými úhly má difúzní složka různou hodnotu a povrch se stává pohledově závislý. Tento model je dnes široce využíván při vykreslování drsných povrchů.



Obr. 2.8. Seskupení odrazu z drsného povrchu. Převzato z [7]

Při výpočtu se bere každý faset jako „Lambertův“ povrch. Výsledné záření odrážející se od povrchu získáme jako:

$$L_r = \frac{\varrho}{\pi} \cdot \cos\theta_i \cdot (A + B \cdot \max(0, \cos(\phi_i - \phi_r)) \cdot \sin\alpha \cdot \tan\beta) \cdot L_i \qquad (2.5)$$

$$A = 1 - 0.5 \frac{\sigma^2}{\sigma^2 + 0.33} \quad B = 0.45 \frac{\sigma^2}{\sigma^2 + 0.09} \quad \alpha = \max(\theta_i, \theta_r) \quad \beta = \min(\theta_i, \theta_r)$$

L_i - zářivost příchozího světla, ϱ - drsnost povrchu, ϱ - albedo povrchu

Obr. 2.9. Porovnání Lambertova a Oren-Nayarova modelu. Převzato z [7]

2.4.3 Blinnův model reflexe

Tato kapitola byla převzata z [8]. Při výpočtu reflexe Phongův model neustále přepočítává úhel **R . V** mezi pozorovatelem a paprskem ze světelného zdroje odraženého od povrchu. Pokud namísto toho vypočteme tzv. „vektor na půl cesty“, který leží mezi vektorem k pozorovateli a vektorem ke zdroji světla, můžeme nahradit

$$\mathbf{R} \cdot \mathbf{V} \qquad (2.6)$$

za

$$\mathbf{N} \cdot \mathbf{H}, \qquad (2.7)$$

kde **N** je normalizovaná normála povrchu.

Úhel získaný pomocí vzorce (2.7) je sice menší než požadovaný úhel ve Phongově modelu, ale jelikož Phong počítá s

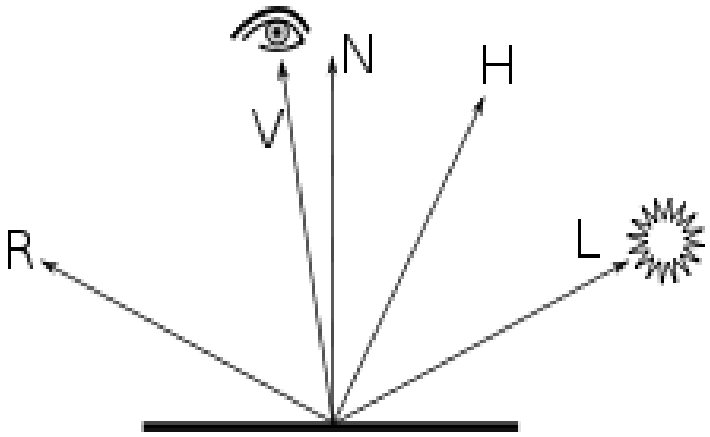
$$(\mathbf{R} \cdot \mathbf{V})^{\alpha}, \qquad (2.8)$$

tak existuje takový exponent **α´**, kde vyhodnocení výrazu

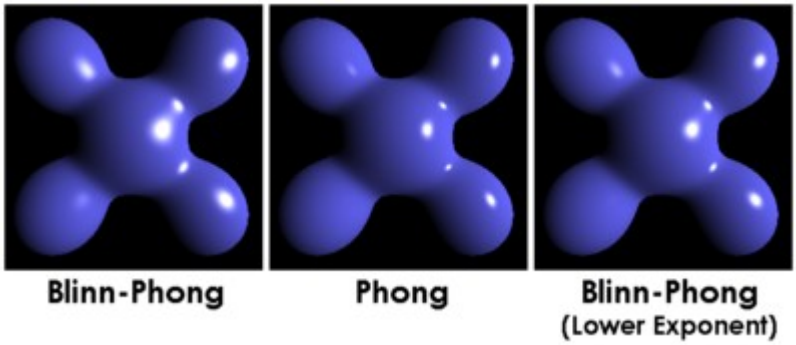
$$(N \cdot H)^{\alpha'} \tag{2.9}$$

se velmi blíží požadované hodnotě.

Úprava, kterou Jim Blinn provedl na Phongově modelu reflexe, nám značně urychlí výpočet, neboť vzorec (2.7) není závislý na poloze a zakřivení povrchu. Stačí jej tedy vypočíst pro každý světelný zdroj pouze jednou.



Obr. 2.10. Vektory pro výpočet Phongova a Blinn-Phongova modelu.
Převzato z [8]



Obr.2.11. Porovnání výsledků Phongova a Blinn-Phongova modelu. Převzato z [8]

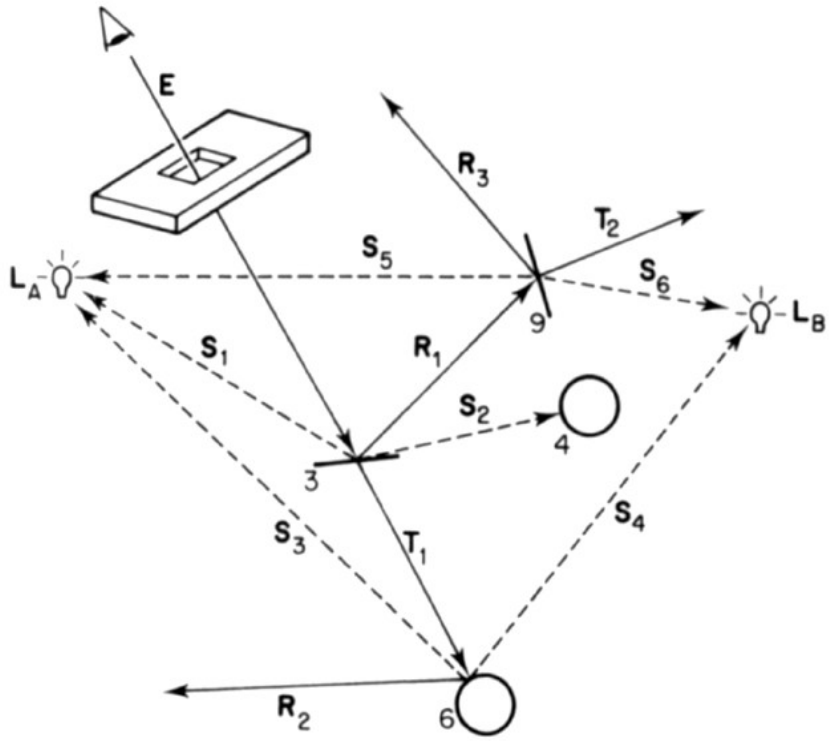
2.4.4 Torrance-Sparrowův model reflexe

Torrance-Sparrowův model se využívá k výpočtu lesklé složky na drsném povrchu. Odvození vzorce provedli dva odborníci z aplikované fyziky Torrance a Sparrow. Jejich metoda ale nebyla vytvořena pro oblast počítačové grafiky. Pro její využití v této oblasti ji upravil R.L. Cook. Metoda vychází z toho, že se povrch skládá z drobných plošek a krystalů. Jak vypadá výpočet Torrance-Sparrowova modelu je detailně rozebráno v [9].

2.4.5 Rekurzivní sledování paprsku

Máme definovanou scénu se zdroji světla a různými objekty. Nyní chceme zjistit, jak bude tato 3D scéna promítnuta na 2D průmětnu. Jak určíme barvu jednotlivých pixelů projekční roviny? Vyšleme paprsek do scény a nalezneme průsečík s nejbližším objektem. Z bodu tohoto průsečíku posíláme stínové paprsky, odrazové a lomené. Pro stínové paprsky, které úspěšně dorazily ke zdroji světla vypočteme osvětlovací model. K výsledné barvě přičteme barvu odražených a lomených paprsků, násobenou koeficientem k<1. Tím, že přičítáme jen určitý poměr u těchto složek, zajistíme útlum při odrazu paprsku.

Odražené a lomené paprsky zjistíme rekurzivním sledováním. Stávají se z nich primární paprsky a aplikujeme na ně stejnou proceduru jako na paprsek vyslaný od pozorovatele. Jelikož používáme rekurzy při vyhodnocení jednotlivých paprsků, musíme si nadefinovat podmínky, při kterých rekurzy zastavíme. Nejčastěji se používá k zastavení rekurze opuštění paprsku scény, dosáhnutí hranice, kdy je již příspěvek barvy příliš malý nebo pokud počet kroků přesáhne pevnou hloubku rekurze.



Obr. 2.12. Rekurzivní sledování paprsku. Převzato z [4]

3 Sledování paprsku a nalezení průsečíků s objekty

Hlavní myšlenkou při hledání průsečíků s objekty je vložení matematické rovnice vyjádřující paprsek do rovnice popisující objekt a určení zda existuje reálné řešení. Pokud existuje, pak existuje i průsečík. Ve většině případů existuje více průsečíků a naším úkolem je vrátit ten nejbližší. V případě stínových paprsků zjišťujeme, zda existuje průsečík se nějakým objektem, který je blíže než průsečík se zdrojem světla. Jestliže testujeme protnutí obalových těles, pak nám stačí pouze znalost o jeho existenci.

3.1 Vyjádření paprsku

Paprsek je definován svým počátkem **O** a směrem **D**. Počátek u primárních paprsků je oko pozorovatele, u sekundárních paprsků je jím bod, z kterého byly vyslány. Směr paprsku je definován tzv. offsetovým vektorem. Rovnice paprsku vypadá následovně:

$$\mathbf{R=O+tD} \tag{3.1}$$

Při snaze nalezení průsečíku s objektem hledáme nejmenší kladnou hodnotu **t**.

3.2 Primitiva

Primitiva jsou objekty, které jsou snadno matematicky popsatelné. Nejčastěji máme na mysli kouli, krychli, kvádr, válec, kužel a podobné. Z takovýmito objekty se poměrně snadno hledá průsečík. Vezmeme jejich matematické vyjádření a vložíme do něj rovnici paprsku. Poté hledáme takovou kladnou vzdálenost na paprsku, která odpovídá průsečíku s primitivem. Je-li hodnota vzdálenosti záporná, primitivum se nachází za počátečním bodem paprsku a paprsek jej tedy nemůže protnout.

Samotný průsečík nalezneme tak, že do rovnice paprsku dosadíme vzdálenost, kterou jsme nalezli. Musíme mít ale na paměti, že vlivem konečné přesnosti počítače průsečík nemusí ležet přesně na povrchu objektu. Tím pak vznikají problémy při vrhání sekundárních paprsků, neboť můžeme detekovat falešný průsečík se stejným povrchem v jeho bezprostřední blízkosti.

Takovýto problém můžeme řešit několika způsoby. Můžeme vyloučit stejné těleso ze zpracování sekundárních paprsků. Což ale vede ke určitým problémům. Toto řešení lze použít jen na stínové a odražené paprsky a navíc těleso samo na sebe nemůže vrhat stín ani se v sobě zrcadlit.

Daleko lepším řešením je posunout průsečík ve směru normály ven z tělesa pro stínové a odražené paprsky. Pro lomené paprsky se pak průsečík posouvá dovnitř. Toto řešení neklade žádné omezení na další zpracování.

Nyní si ukážeme, jak vypadá nalezení průsečíku paprsku s koulí. U ostatních primitiv se využívá stejného postupu. Mějme kouli s poloměrem 1 umístěnou v počátku souřadnic. Tuto kouli můžeme vyjádřit pomocí:

$$x^2+y^2+z^2=1 \tag{3.2}$$

Po dosazení rovnice paprsku do rovnice koule získáme:

$$(O_X+tD_X)^2+(O_Y+tD_Y)^2+(O_Z+tD_Z)^2=1 \tag{3.3}$$

To lze vyjádřit jako:

$$t^2(D_X^2+D_Y^2+D_Z^2)+t(2O_XD_X+2O_YD_Y+2O_ZD_Z)+(O_X^2+O_Y^2+O_Z^2-1)=0 \tag{3.4}$$

Na první pohled je zřejmé, že se jedná o kvadratickou rovnici:

$$at^2+bt+c=0 \tag{3.5}$$

Vyřešením této kvadratické rovnice dostaneme dvě vzdálenosti **t**, které nám udávají průsečíky, kde paprsek do koule vstoupil a kde ji opustil.

3.3 Složitější objekty

V reálné scéně se málokdy setkáme jen s primitivními objekty. Ba naopak, ve spoustě případů mají objekty daleko složitější tvar. Pokud se ale na takové předměty podíváme zblízka, všimneme si lokálních oblastí, které nám svým tvarem primitiva připomínají. Z toho můžeme usoudit, že většina komplexních objektů půjde vytvořit ze základních primitiv za pomoci operací sjednocení, průnik a rozdíl. Takovémuto přístupu na vytváření 3D modelů se říká konstruktivní geometrie.

Jiným přístupem na tvorbu složitějších objektů je tvorba tzv. polygonálních modelů. Polygonální model si můžeme představit jako různě v 3D prostoru poskládané trojúhelníky, které svým uspořádáním popisují tvar povrchu objektu. Hladkost a tedy i realističnost takovéhoto modelu záleží na počtu a velikosti trojúhelníků. Touto metodou lze vytvořit jakýkoliv tvar.

3.4 Povrchové vs. objemové modely

Jak jsme si v předchozí kapitole uvedli, máme několik možností, jak vytvářet složitější objekty. Buď pouze na základě popisu jeho povrchu (polygonální modely), pak se bavíme o povrchových modelech, nebo na základě popisu celého jeho objemu (konstruktivní geometrie), pak mluvíme o objemových modelech. Každý přístup má své výhody i nevýhody.

Polygonální modely jsou nejčastěji využívány v počítačových hrách a v jiných aplikacích, které vykreslují v reálném čase. Jejich velké rozšíření je dáno tím, že grafické karty jsou přímo vytvořeny pro práci s takto popsanými objekty a vykreslení trojúhelníku je v nich velice rychlé. Nevýhodou tohoto přístupu je, že pokud chceme mít dostatečně detailní předmět, musíme použít obrovské množství trojúhelníků pro jeho popis. Pak samotný popis se stává obrovsky složitým a jelikož máme velké množství trojúhelníků, tak i rychlost vykreslení je velmi pomalá. Takto složité modely se ale v aplikacích reálného času nepoužívají.

Pokud vytvoříme model pomocí konstruktivní geometrie, dostaneme poměrně snadno jeho dostatečně kvalitní popis. Problém je jak takovýto model zobrazit. Jednou z metod jak jej zobrazit je právě metoda sledování paprsku. Dalo by se říci, že tyto dvě metody jsou spolu úzce spjaty. Jelikož vykreslování pomocí metody sledování paprsku nepatří mezi nejrychlejší, tak takovéto modely a scény vytvořené pomocí nich se používají v aplikacích, u kterých nezáleží tak na rychlosti vykreslení jako na kvalitě výsledného obrazu.

Volíme-li si metodu, pomocí které budeme vytvářet složité objekty, musíme vždy mít na paměti, v jaké aplikaci chceme tyto objekty použít a jak je chceme zobrazovat. V oblasti metody sledování paprsku se nejčastěji využívá konstruktivní geometrie. Lze použít i polygonálního popisu, ale mějme na paměti, že detailnost modelů takto vytvořených bude záviset na počtu použitých trojúhelníků a že nalezení průsečíku mezi paprskem a trojúhelníkem není nejrychlejší.

3.5 Transformace objektů

Chceme-li si rozšířit škálu objektů, ze kterých je možno vytvářet složitější modely, zavedeme možnost různé transformace nad jednotlivými primitivy nebo dokonce nad samotnými uzly stromu konstruktivní geometrie. Pro transformaci objektů v počítačové grafice se využívají transformační matice. Použití těchto matic je ale v oblasti sledování paprsku trošku odlišné.

3.5.1 Klasický přístup k transformaci objektů

Máme objekt, který chceme transformovat. Tento objekt leží na určitých souřadnicích někde ve scéně. Pokud chceme provést transformace pouze na tomto objektu, musíme jej nejdříve posunout do

počátku souřadnic, pak provedeme požadované transformace a objekt zase vrátíme na jeho původní místo.

Samotná transformace spočívá v tom, že bod, jenž chceme transformovat, vynásobíme transformační maticí.

$$\mathbf{A}_M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix} \qquad \mathbf{A}_{Sc} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Obr. 3.1. Ukázka transformačních matic: Index M – posunutí, Sc – změna velikosti

$$\mathbf{A}_{Rx} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{A}_{Ry} = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{A}_{Rz} = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ -\sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Obr. 3.2. Ukázka transformačních matic: Rotace podle jednotlivých os

$$\mathbf{A}_{Shyz} = \begin{bmatrix} 1 & Sh_y & Sh_z & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{A}_{Shxz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ Sh_x & 1 & Sh_z & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{A}_{Shxy} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ Sh_x & Sh_y & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Obr. 3.2. Ukázka transformačních matic: Zkosení jednotlivých os

3.5.2 Transformace objektů u metody sledování paprsku

Tato kapitola je převzata z [10]. Mějme objekt, který chceme transformovat. Vytvoříme transformační matici tak jako v klasickém přístupu. Ovšem těžko se detekuje průsečík na transformovaném objektu. Musíme tedy transformovat paprsek.

Předpokládejme, že máme transformovaný objekt ve světových souřadnicích **WS** (souřadnicích scény). Hledáme tedy průsečík paprsku s tímto objektem. Jelikož nechceme transformovat objekt ale paprsek, pak musíme najít takovou pozici a směr paprsku v souřadnicích objektu **OS**, které odpovídají pozici a směru ve světových souřadnicích, při nalezení stejného bodu. Toto provedeme tak, že vytvoříme inverzní matici k transformační matici a touto maticí upravíme paprsek (3.2) následovně:

$$\mathbf{O}_{os} = \mathbf{M}^{-1} \cdot \mathbf{O}_{ws} \tag{3.6}$$

$$\mathbf{D}_{os} = \mathbf{M}^{-1} \cdot \mathbf{D}_{ws} \tag{3.7}$$

Je důležité zmínit, že každá složka paprsku se transformuje jiným způsobem. Jak kterou složku transformovat vidíme na obrázku 3.3 a 3.4.

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} ax+by+cz+d \\ ex+fy+gz+h \\ ix+jy+kz+l \\ 1 \end{pmatrix}$$

Obr. 3.3. Transformace výchozího bodu paprsku. Převzato z [10]

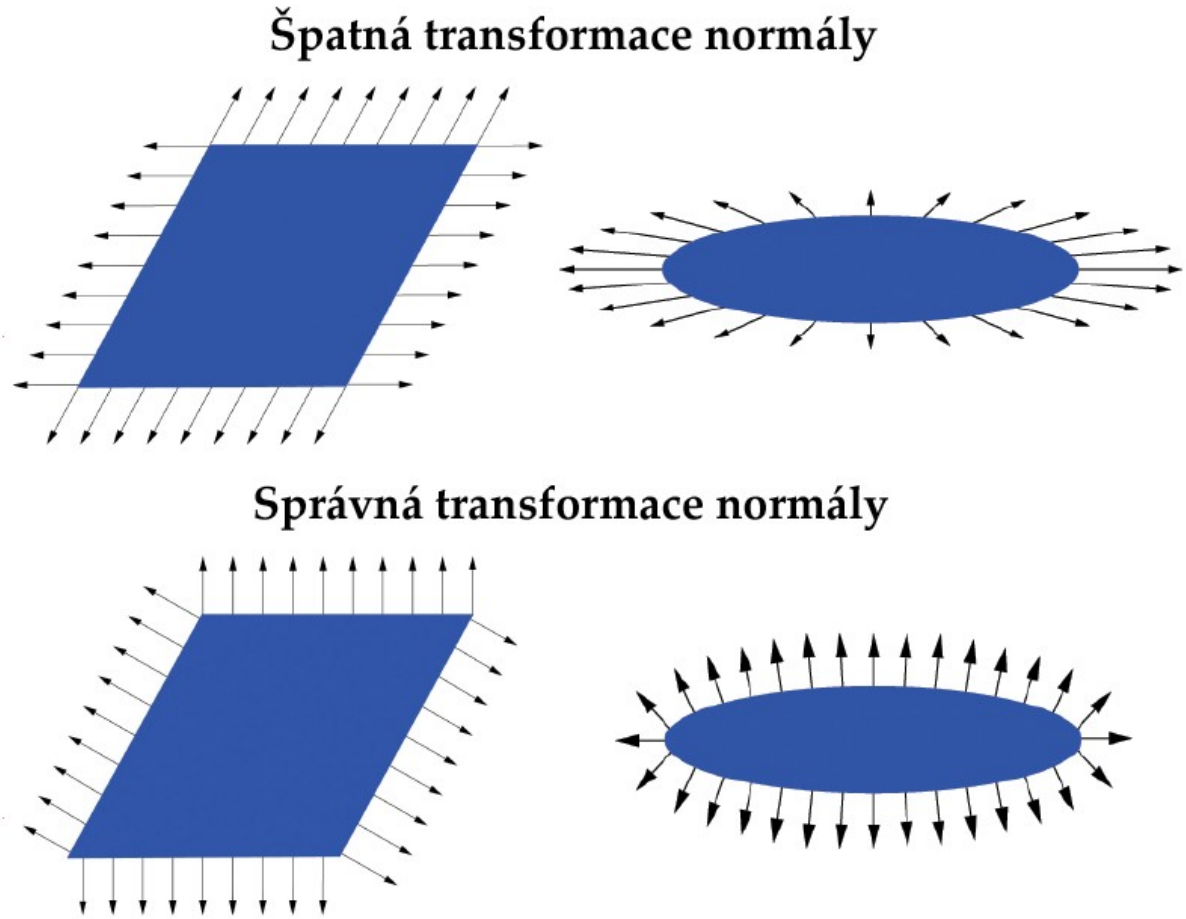
$$\begin{pmatrix} x' \\ y' \\ z' \\ 0 \end{pmatrix} = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 0 \end{pmatrix} = \begin{pmatrix} ax+by+cz \\ ex+fy+gz \\ ix+jy+kz \\ 0 \end{pmatrix}$$

Obr. 3.4. Transformace směru paprsku. Převzato z [10]

Také nám vzniká jeden problém. Pokud transformace zahrnuje změnu velikosti, pak směr paprsku v souřadnicích objektu nebude normalizován. Když jej normalizujeme, vzdálenost průsečíku na paprsku v souřadnicích objektu nebude totožná se vzdáleností, kterou bychom našli ve světových souřadnicích a musíme tuto vzdálenost upravit po nalezení průsečíku. Pokud jej nenormalizujeme, pak sice vzdálenost bude stejná, ale vznikají jiné problémy při hledání průsečíku.

Dále nesmíme zapomenou, že i normála bude transformovaná. Abychom dosáhli správné transformace, musíme normálu upravit podle následujícího vzorce:

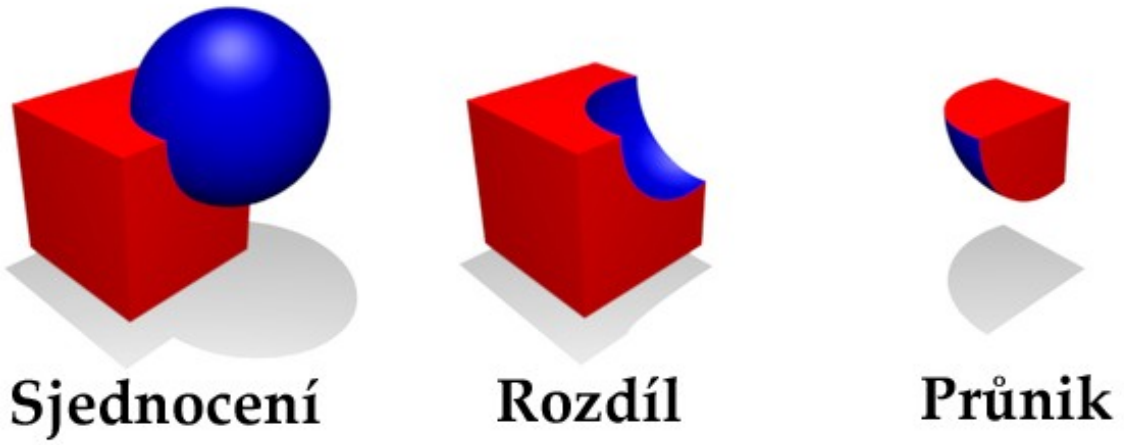
$$\mathbf{N}_{ws} = (\mathbf{M}^{-1})^{T} \cdot \mathbf{N}_{os} \tag{3.8}$$



Obr. 3.5. Znázornění rozdílů mezi transformací normály jen za použitím transformační matice a při použití správného vzorce. Převzato z [10]

4 Konstruktivní geometrie

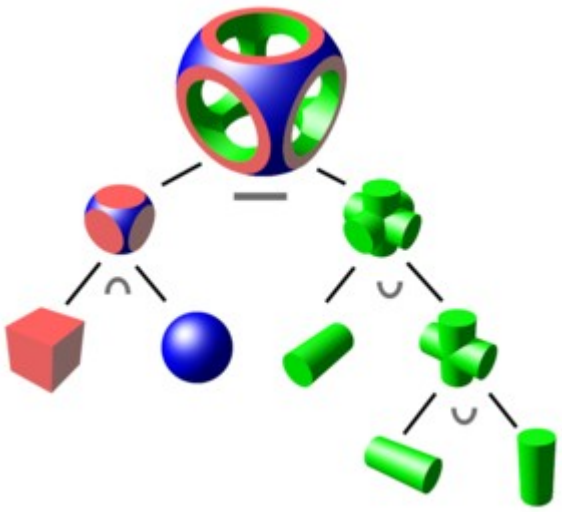
Konstruktivní geometrie, také známá pod zkratkou CSG (z angl. Constructive Solid Geometry), je metoda využívající se k tvorbě složitých 3D modelů. Jejich reprezentace je založena na analytickém popisu těles pomocí jejich objemu. Díky tomu, že nemáme objekt popsán jen jeho hraniční reprezentací, ale známe celý jeho objem, můžeme si dovolit nad jednotlivými objekty provádět množinové operace. To nám umožňuje ze dvou těles definovat těleso nové. Tato vlastnost je velmi důležitá, neboť nám dovoluje popsat relativně komplexní objekt pomocí triviálních snadno popsatelných objektů, nad nad nimiž jsou operace prováděny.



Obr. 4.1. Ukázka operací s primitivy. Převzato z [11]

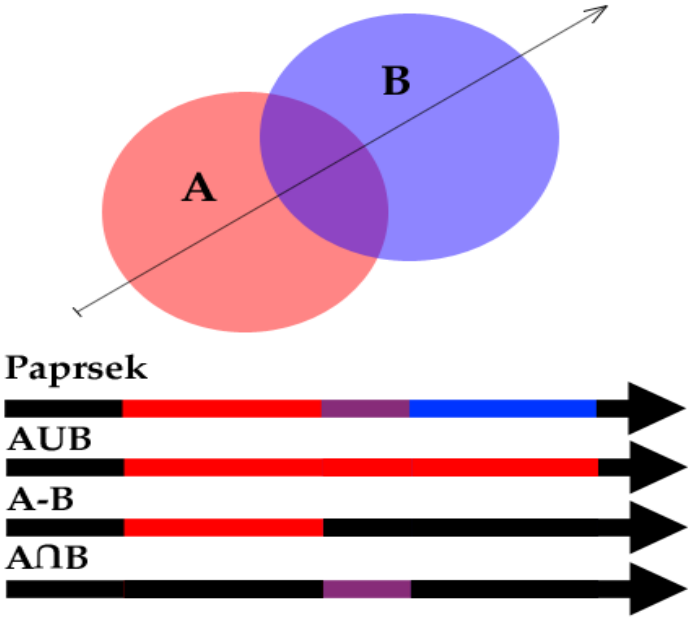
Základním stavebním kamenem pro konstruktivní geometrii jsou primitiva. Jedná se o objekty, které jsou snadno popsatelné pomocí jedné nerovnice nebo pomocí soustavy několika nerovnic. Mezi primitiva patří například: koule, kvádr, válec, kužel atd.

Složité těleso je tedy popsáno pomocí stromu, který obsahuje ve svých listech geometrické informace o primitivních tělesech. V uzlech tohoto stromu se nacházejí množinové operace, popřípadě i transformace, které jsou nad jednotlivými synovskými uzly prováděny. Mezi tyto operace patří sjednocení, průnik a rozdíl.



Obr.4.2. Ukázka CSG stromu. Převzato z [11]

Základní myšlenkou pro zobrazení takto vzniklých 3D modelů za pomoci sledování paprsku je nalezení intervalů na vrženém paprsku, které náleží jednotlivým primitivům uložených v listech stromu. Nad těmito intervaly jsou pak prováděny množinové operace definované CSG stromem. V kořenovém uzlu pak dostáváme intervaly na paprsku, které náleží komplexnímu modelu a tedy známe i průsečíky s tímto modelem. Obdobně jako u zjištění průsečíku s jednoduchými objekty i zde nás zajímá nejmenší kladná vzdálenost na paprsku.



Obr. 4.3. Znázornění operací na paprsku

5 Urychlování metody sledování paprsku

Máme-li mnoho objektů ve scéně, pak klasický přístup metody je velmi pomalý a je potřeba jej vylepšit. Urychlení algoritmu lze provést v několika oblastech. Základem je rychlejší výpočet průsečíků. Využíváme efektivnější rutiny pro výpočet každého průsečíku. Výpočty, které není nutné provádět při každém testování objektu a stačí je vypočíst pouze jednou si dopředu předpočítáme. Ostatní výpočty pak se snažíme provádět s minimálním počtem operací a upřednostňujeme takové operace, které jsou procesorem vyhodnoceny za kratší dobu.

5.1.1 Rychlejší výpočet průsečíků

Jednou z metod urychlování je snížení počtu objektů, s kterými je nutno provádět výpočet průsečíku. K tomu se nejčastěji používají obalová tělesa. Pokud máme složitý objekt, například vytvořený pomocí konstruktivní geometrie, je daleko výhodnější nejprve testovat, zda paprsek protne jednodušší objekt, který jej svými rozměry a pozicí obklopuje. Je-li obalové těleso protnuto, pak se teprve hledá průsečík s komplexnějším objektem. V případě pozitivního nálezu průsečíku máme sice o jeden test navíc, ale pokud paprsek objekt neprotne, pak si ušetříme daleko složitější výpočet, který by byl zbytečný. Mezi nejčastější obalovací tělesa patří koule a kvádr.

Další možností jak snížit počet testovaných objektů je dělení prostoru. Scénu bereme jako jeden velký prostor, který rozdělíme do menších podprostorů. Asi nejvhodnější metodou pro dělení 3D prostoru je pomocí hierarchické stromové datové struktury známé jako oktalový strom (angl. Octree). Každý interní uzel této struktury má až osm dětí, které znázorňují podprostory daného uzlu. Problém nalezení průsečíku paprsku s objektem pak rozdělíme na dvě části. Nejprve hledáme průsečík paprsku s jednotlivými uzly z oktalového stromu a zjišťujeme, který uzel je ve směru paprsku nejblíže. To děláme tak dlouho, dokud nenalezneme nejbližší list. Paprsek pak testujeme jen s objekty, které se v tomto podprostoru nacházejí.

Než začneme využívat jakékoliv dělení prostou, je ale nutné si uvědomit, že ne vždy se nám díky němu vykreslování scény urychlí. Pokud máme scénu jen s pár objekty nebo pokud hledání průsečíku s objekty je výrazně rychlejší než testování protnutí prostoru (tedy hledání průsečíku s kvádrem), pak by nebylo moudré dělení prostoru využívat. Nejen, že v takovém případě by se nám vykreslování scény neurychlilo, ale naopak bychom jej ještě více zpomalili.

5.1.2 Menší počet paprsků

Urychlit algoritmus můžeme také, pokud snížíme počet paprsků, které vyhodnocujeme. Jedna z metod, která se k tomuhle využívá, se nazývá adaptivní vzorkování. Máme průmětnu, která je složena z jednotlivých pixelů. Při klasické metodě sledování paprsku posíláme paprsek skrze všechny pixely průmětny a vyhodnocujeme jeho barvu. Pokud vykreslujeme obrázek, jehož velkou část zabírá pozadí, to znamená že spousta paprsků nemá průsečík s žádným z objektů scény, pak v hodně případech provádíme vyhodnocování paprsku zbytečně. Při adaptivním vzorkování se průmětna rozdělí na menší oblasti, u kterých se vyhodnocuje paprsek jen v jejich rozích. Neprotne-li žádný z těchto paprsků scénu, lze usoudit, že i paprsky vyslané do zbývajících pixelů uvnitř této oblasti scénu neprotnou. Můžeme tedy celou oblast vyplnit barvou pozadí. Pokud se paprsky v rozích liší, tak můžeme buď oblast dále dělit nebo v případě, že je dostatečně malá, poslat paprsky skrze zbývající pixely. Na jak velké oblasti rozdělíme průmětnu závisí na vlastnostech scény.

Tato metoda má i jisté nevýhody. Pokud špatně zvolíme velikost jednotlivých oblastí, může se nám stát, že existují dostatečně malé objekty, které se celé zobrazí uvnitř jedné z těchto oblastí a my jej vůbec nezaznamenáme. V takovém případě jej nezobrazíme.

Adaptivního vzorkování lze využít i pro případné upřednostnění některého z objektů, u něhož je předpoklad, že průsečík s ním bude nakonec vyhodnocen jako nejbližší.

5.1.3 Paralelizace výpočtu

Metoda sledování paprsku je ideální pro paralelní zpracování. Barva každého pixelu se vyhodnocuje zvlášť a není závislá na ostatních pixelech. Vyhodnocení jednotlivých sekundárních paprsků také není na sobě závislé. Můžeme paralelizovat i hledání průsečíků s různými objekty a pak pouze určit, který z nich se nachází nejblíže.

Metoda sledování paprsku je velmi výpočetně náročná. Rozložením výpočtů mezi různé procesory nám velmi zkrátí dobu vykreslování scény.

6 Výhody a nevýhody klasického řešení

Používáme-li klasické řešení, pak dostaneme relativně pěkný obrázek v přiměřeném čase. Ale klasický přístup má jistá omezení. Umožňuje pouze bodová světla. Vytváří pouze tvrdé stíny. Neumí zpracovat průsvitnost. Všechny objekty scény jsou buď neprůhledné nebo dokonale průhledné. Nelze uspokojivě vytvořit lesklé materiály. Máme vždy dokonalý odraz. V obraze vzniká aliasing. Chybí iluze hloubky ostrosti a rozmazání pohybujících se objektů. Nedokáže uspokojivě simulovat všechny vlastnosti světla.

Pokud chceme získat realističtěji vypadající scénu, tak musíme základní metodu upravit. Můžeme například místo jednoho paprsku jich do scény vrhnout více a výslednou barvu získat průměrováním. Takovéto úpravě se říká distribuované sledování paprsku.

7 Distribuované sledování paprsku

Distribuované sledování paprsku nám umožňuje vytvářet fotorealističtěji vypadající scény. Základní myšlenkou je nahradit jeden paprsek svazkem paprsků. Jednotlivé paprsky nesou informace o barvě. Výsledná barva celého svazku je pak dána průměrováním. Díky tomu, že místo jednoho paprsku jich do scény vrháme několik, dosáhneme daleko lepších výsledků. V klasické variantě při vrhání jednoho paprsku jsme vždy dostali jednoznačný výsledek. Trefili jsme objekt nebo netrefili. Vrhá na tento bod nějaký předmět stín nebo nevrhá. Hranice mezi vždy dvěma možnostmi byla velmi zřetelná. Proto jsme mohli dosáhnout pouze tvrdých stínů. Proto také okraje objektů byly ve vykresleném obrázku zubaté.

7.1 Odstranění aliasingu v obraze

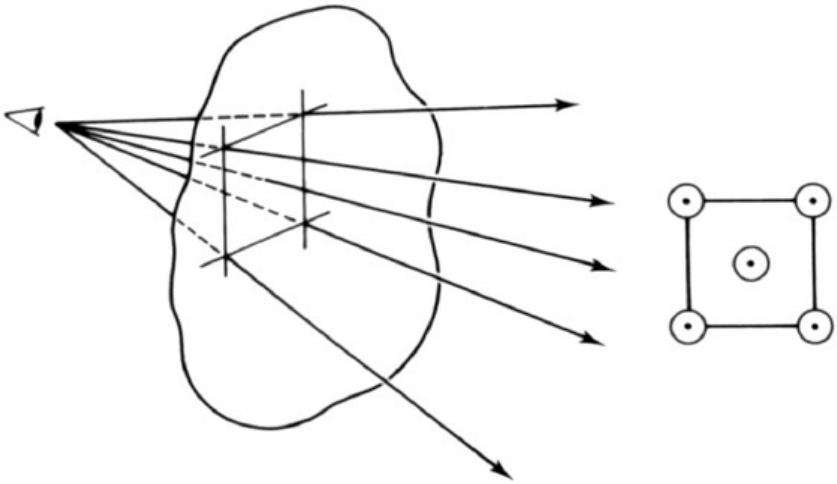
Tradiční systémy sledování paprsku trpí aliasingovými artefakty. Termín „aliasing“ není v počítačové grafice striktně definován. Tento termín může znamenat téměř cokoliv nežádoucího v obraze. Typicky se používá k popisu zubatého okraje. V reálném světě nejsou věci tak dokonalé jako v počítačově vygenerované scéně. Hrany nejsou ostré, odrazy nejsou dokonalé. Aliasingový artefakt lze jednoduše odstranit tím, že použijeme distribuované sledování paprsku. Konkrétně je odstraněn vrhnutím svazku primárních paprsků [12].

Při klasickém sledování paprsku jsme barvu pixelu průmětny zjišťovali tak, že jsme skrze jeho střed poslali paprsek do scény. Jelikož v distribuovaném sledování paprsku nemámě paprsek jenom jeden, ale několik, pak vzniká otázka, jak jsou tyto paprsky ve svazku uspořádány, neboli jakým směrem budeme jednotlivé paprsky posílat.

Možností máme hned několik. Předpokládejme, že máme svazek o pěti paprscích. Můžeme tedy každým pixelem vrhnout jeden paprsek středem a ostatní skrze jeho rohy. Pokud chceme mít výslednou barvu pixelu co nejpřesnější, pak nemusíme tento počet paprsků brát za konečný a můžeme si říci, že pokud výsledky, které jsem získaly od jednotlivých paprsků, se výrazně liší, pak pixel rozdělíme na další 4 oblasti a do každé z nich pošleme nových 5 paprsků. Takto můžeme pokračovat, dokud nebudeme spokojeni s výsledkem. Takovéto metodě říkáme adaptivní nadvzorkování [4]. Výhodou této metody je, že nemusíme posílat konstantní velké množství paprsků, abychom dosáhli co nejpřesnější barevné representace daného pixelu. Nevýhodou však je, že pokud existuje dostatečně malý (nebo dostatečně vzdálený) objekt, tak jej díky tomuto rozložení nemusíme zasáhnout a barvu pixelu ohodnotíme, jako by v dané oblasti vůbec neexistoval.

Jinou možností je náhodně vygenerovat pozice uvnitř pixelu, které nám budou určovat, jak bude svazek paprsků rozložen. Tato varianta je daleko lepší než používat předem nadefinované

pozice. I tuto variantu lze upravit, aby efektivně určila co nejpřesnější výslednou barvu. Vybereme si konstantní počet paprsků, které vrháme do každého z pixelů. Pokud se jednotlivé barvy uvnitř pixelu výrazně lišší od průměrné hodnoty, tak přidáme další paprsky a znovu vypočteme průměrnou hodnotu. To lze aplikovat tak dlouho, dokud výsledek v daném pixelu nesplňuje naše požadavky.



Obr. 7.1. Adaptivní nadvzorkování. Převzato z [4]

Posíláme-li všemi pixely svazky o stejném rozložení, pak se sice zbavíme aliasingu, ale nastává jiný problém. Výsledný obraz je ovlivněn mřížkou, která vznikla použitím pravidelného uspořádání paprsků. Abychom tomuto zabránili, pak musíme použít v každém pixelu jiné rozložení. Nejjednodušší variantou je pro každý pixel průmětny vygenerovat nové náhodné pozice.

7.2 Možnost nových efektů

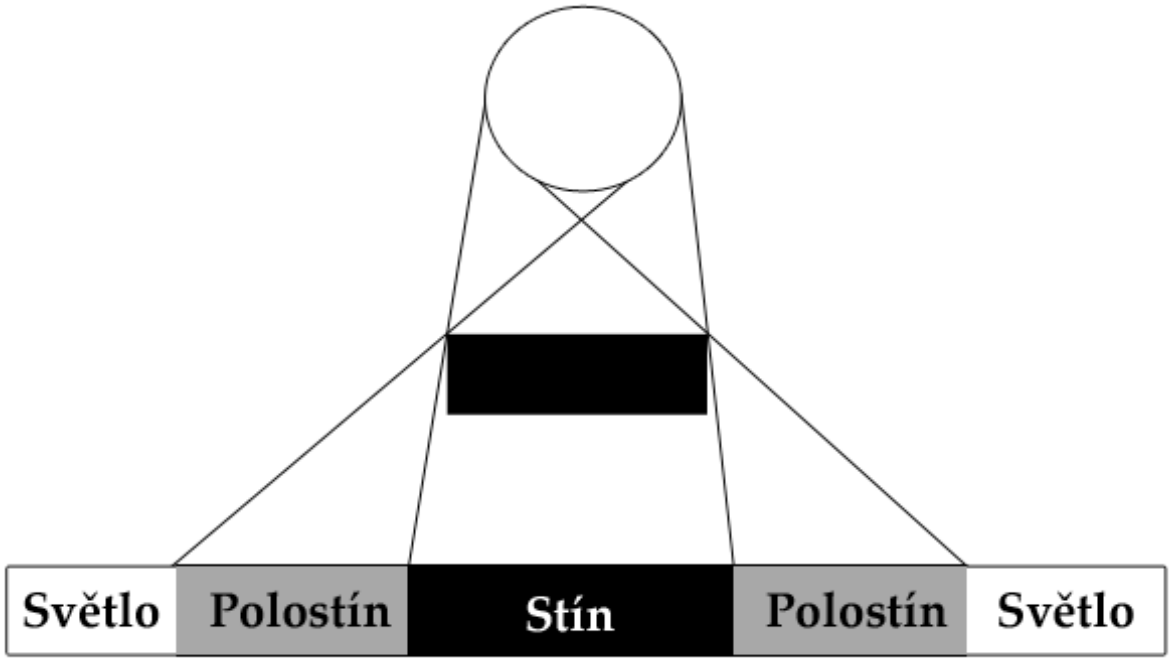
Jak jsme si popsali výše, poslání svazku paprsků je daleko výhodnější než poslání paprsku samotného. Proto toho metoda distribuované sledování využívá nejen u primárních, ale i u sekundárních paprsků. Tím docílíme různých efektů.

7.2.1 Jemné stíny

Stíny v tradiční metodě sledování paprsku jsou diskrétní. Chceme-li určit, zda se daný bod nachází ve stínu, pak vyšleme stínové paprsky ke všem zdrojům světla. Každý z viditelných zdrojů pak přispívá ke stínování. Světla sami o sobě jsou modelovány jedním bodem, což je docela přesná reprezentace zdrojů, které jsou umístěny daleko od nás, ale velmi nepřesná vzhledem ke zdrojům, které jsou velké nebo se nacházejí v blízké vzdálenosti. Důsledkem diskrétního rozhodování je zřetelný okraj u

jednotlivých stínů. V reálném světě jsou stíny mnohem měkčí. Děje se tak díky omezené oblasti světelných zdrojů a díky rozptylu světla z jiných povrchů.

Problematika jemných stínů nám rozděluje body na povrchu do tří základních oblastí. Na oblast beze stínu, polostín a stín. S body, které jsou uvnitř oblasti beze stínu nebo uvnitř oblasti plného stínu, se nám pracuje poměrně snadno. Menší problém nám vytvářejí body uvnitř polostínu. Otázkou je, jak efektivně určit, kde se daný bod v polostínu nachází.

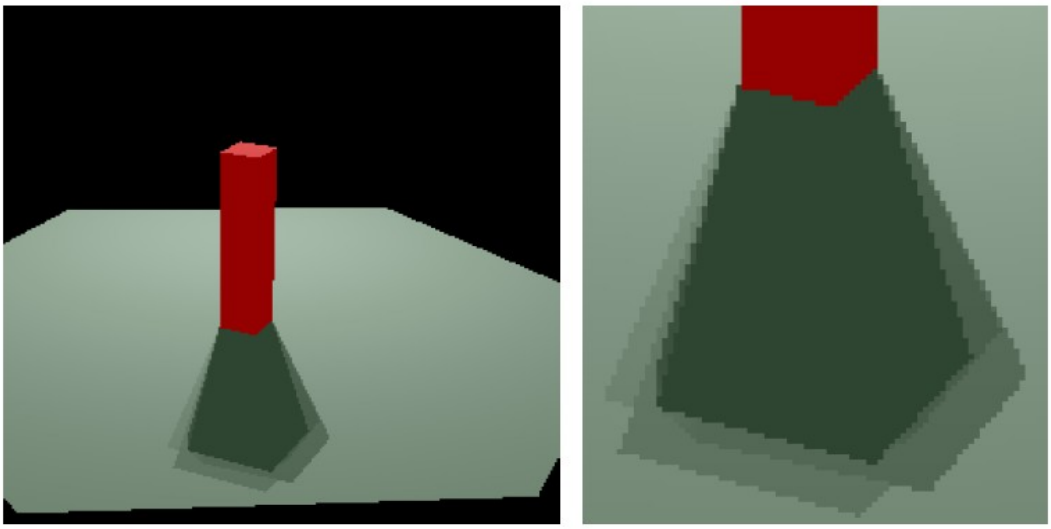


Obr. 7.2. Znázornění problematiky měkkých stínů

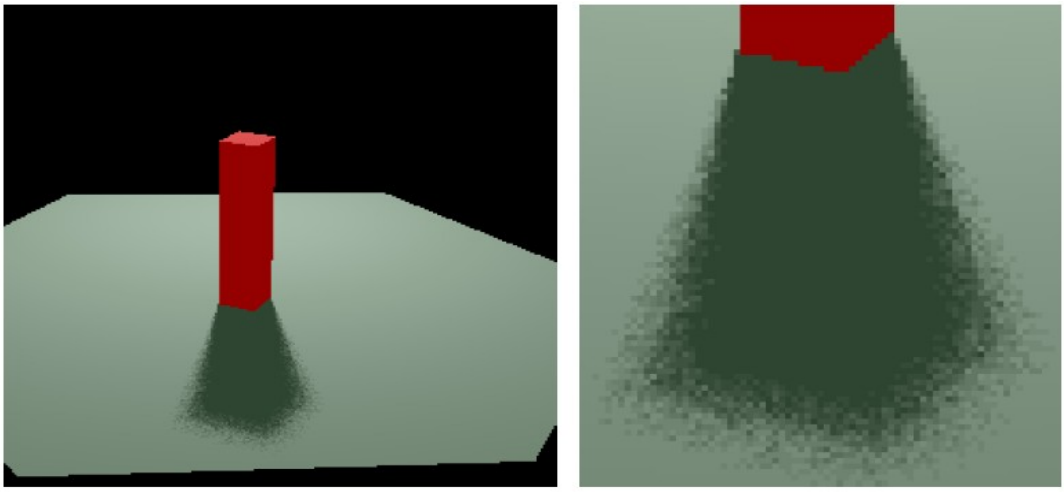
Pokud se chceme k jemným stínům přiblížit, tak nejprve musíme změnit reprezentaci světelných zdrojů. Popis pomocí bodů nám již nebude stačit. Každý zdroj bude nyní reprezentován pomocí nějakého tělesa. Nejčastěji se používá koule, ale není to podmínkou.

Další úprava spočívá v použití metody Monte Carlo pro vyhodnocení stínu. Náhodně se na povrchu světelného zdroje vygenerují body, které nám slouží jako cíle jednotlivých paprsků. Vyšleme stínové paprsky a testujeme, kolik se jich dostalo k světelnému zdroji. Prošlé paprsky nám udávají míru světla, kterým je bod daným zdrojem osvětlen.

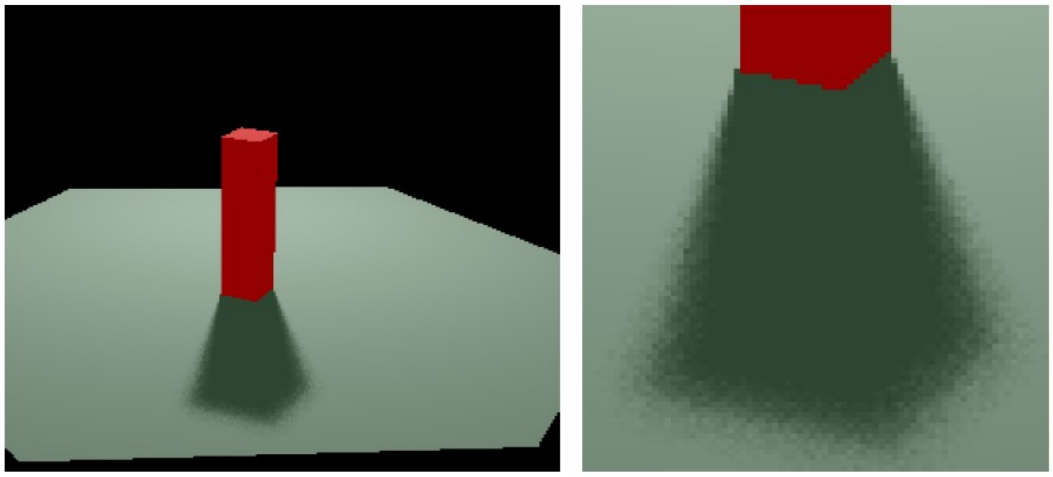
Tato metoda má ale jednu nevýhodu. Chceme-li dosáhnout kvalitně vypadajícího stínu, musíme vyslat velké množství paprsků. To nám výrazně zpomaluje výpočet celého algoritmu. Proto je snaha o vyslání menšího počtu paprsků a nějakým způsobem oblast polostínu dopočíst. Existuje několik metod, které řeší tuto problematiku. Mezi ty hlavní patří metoda „Polostínového klínu“ (angl. Penumbra wedges) a metoda „Těles měkkého stínu“ (anlg. Soft shadow volumes). Tyto řeší problém jemného stínu s velmi uspokojivými výsledky.



Obr.7.3. Jemný stín – uniformní vzorkování. Převzato z [4]



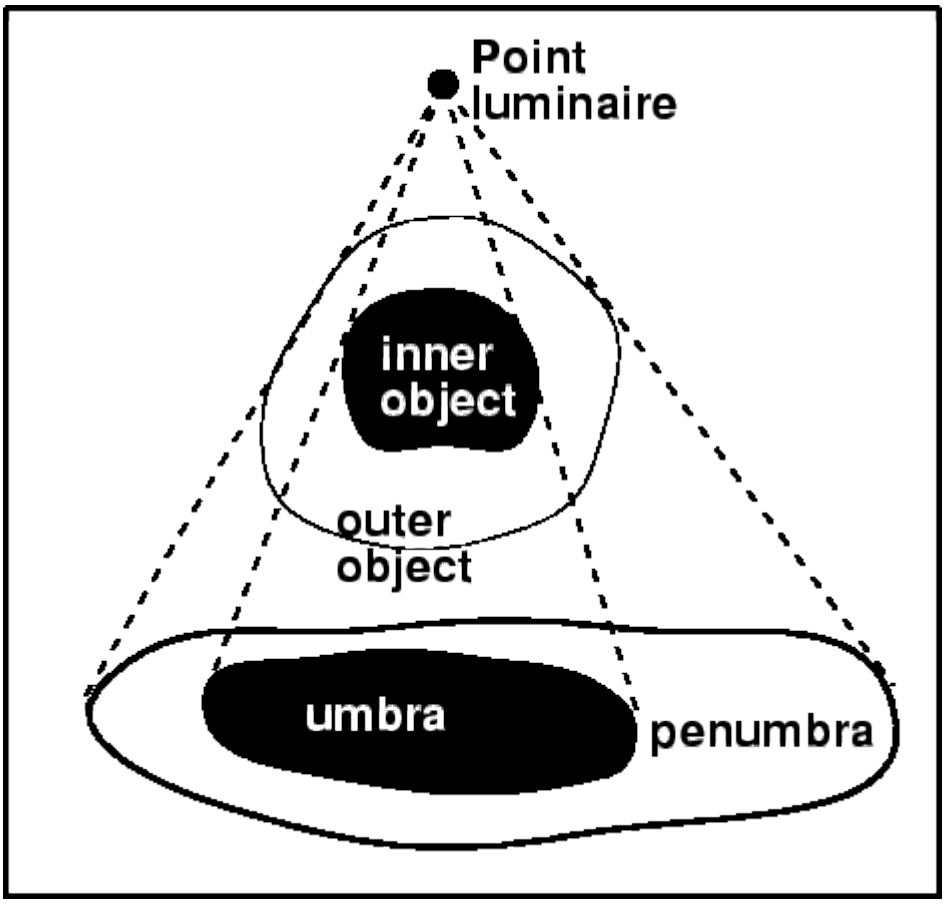
Obr. 7.4. Jemný stín – distribuované sledování paprsku (10 paprsků). Převzato z [4]



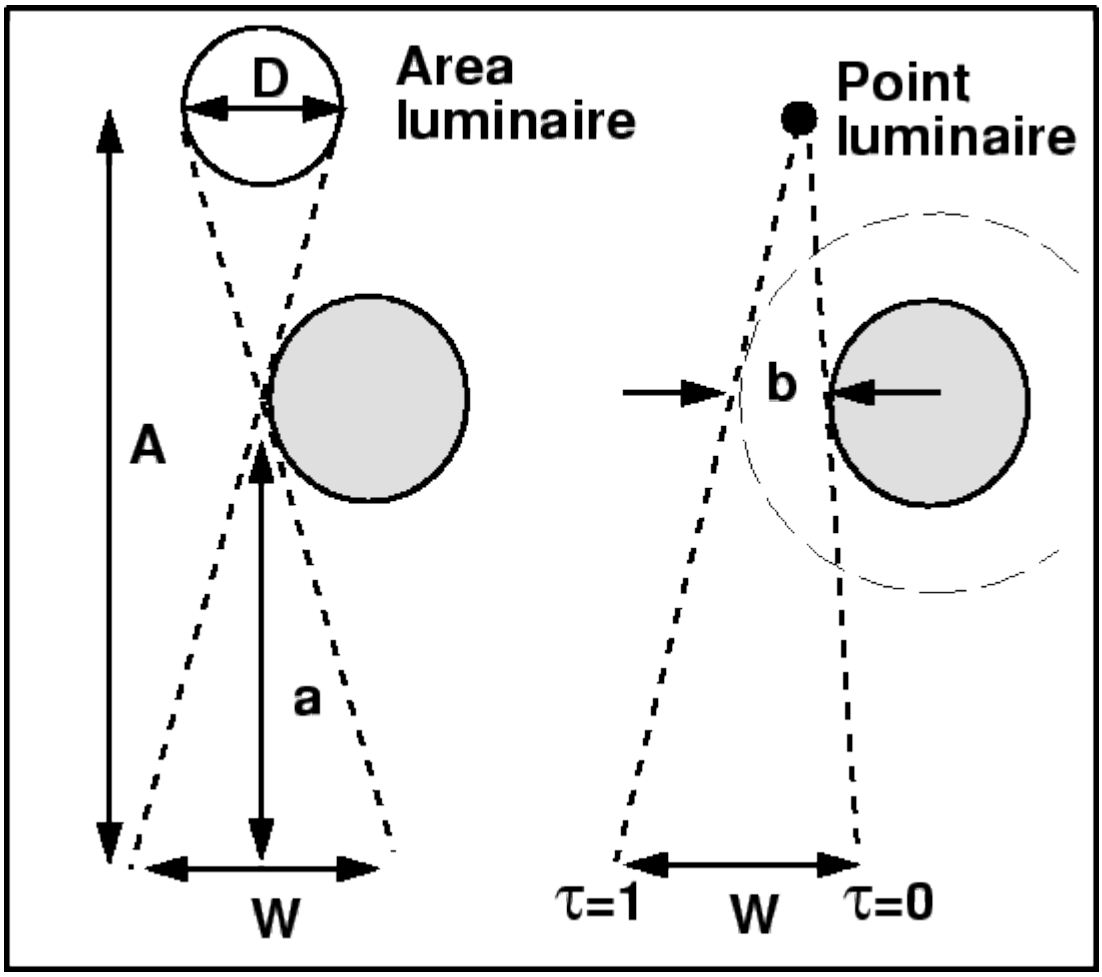
Obr. 7.5. Jemný stín – Distribuované sledování paprsku (50 paprsků). Převzato z [4]

Nejzajímavější je ale asi metoda S. Parkera, P. Shirleyho a B. Smitse zvaná „Měkký stín jedním vzorkem“ (angl. Single Sample Soft Shadow). Hlavní myšlenkou této metody je vrhnutí jednoho paprsku směrem ke zdroji světla a nalezení bodu na tomto paprsku, který je k objektu vrhající stín nejblíže. Podle vzdálenosti tohoto bodu od objektu se pak určí hodnota stínu.

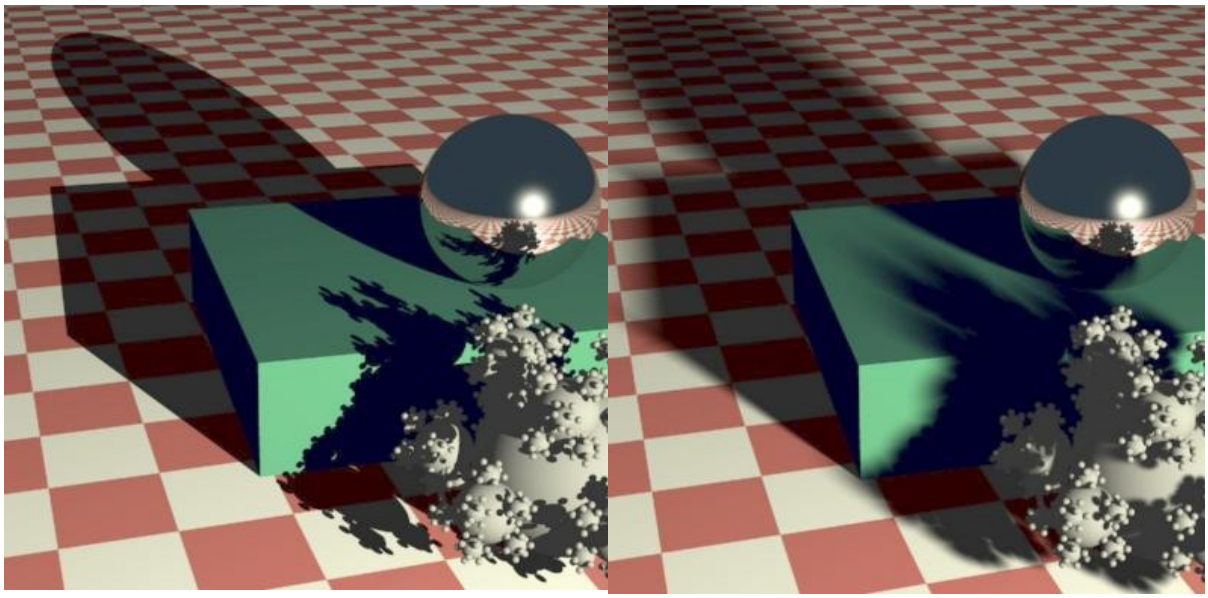
Metoda předpokládá použití koulí jako zdrojů světla. Na určení, v jaké oblasti stínu se daný bod nachází, se využívají dvě tělesa. Oblast stínu znázorňuje objekt samotný, oblast polostínu pak zvětšenina tohoto objektu. Jak moc zvětšit objekt nám určuje vzdálenost nejbližšího bodu na paprsku a velikost zdroje světla. K určení polostínu vrhneme paprsek směrem ke středu světla a hledáme takový bod na tomto paprsku, který je nejblíže stínovému tělesu. Podle tohoto bodu pak určíme hodnotu stínu. Nachází-li se uvnitř objektu, který vrhá stín, pak máme stín plný. Nachází-li se mimo objekt znázorňující polostín, pak nemáme stín žádný. Třetí možností je, že bod se nalézá uvnitř polostínového tělesa. Pak se určí jeho procentuální umístění vzhledem k šířce oblasti polostínu. Pomocí něj pak vypočteme hodnotu polostínu, která je stínovým objektem vrhána [13].



Obr.7.6. Ukázka, jak se tvoří oblast polostínu. Převzato z [13]



Obr.7.7. Ukázka určení šíře polostínového tělesa. Převzato z [13]

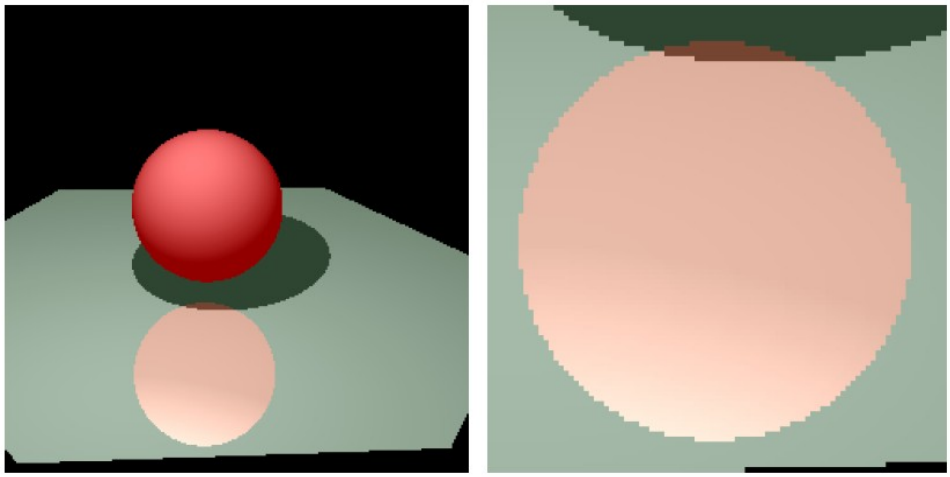


Obr. 7.8. Výsledek metody „Měkkého stínu jedním vzorkem“. Převzato z [13]

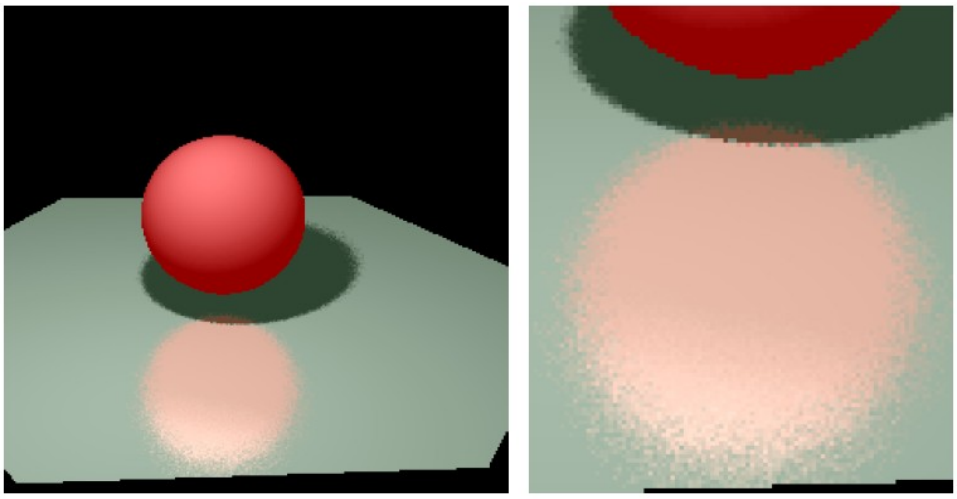
7.2.2 Lesk

Tato kapitola byla převzata z [12]. Tradiční metoda je dobrá v reprezentaci dokonale odrážejících ploch. Ve skutečnosti ale identickou scénu odrážejí jen perfektní zrcadla. V reálném světě se daleko častěji setkáváme s objekty, které jsou lesklé a které odrážejí rozmazaný obraz scény. Je tomu tak díky rozptylu světla. Reprezentace takových materiálů je v klasické metodě problém. Odrazy v klasické metodě jsou vždy ostré, i když se jedná třeba jen o částečný odraz.

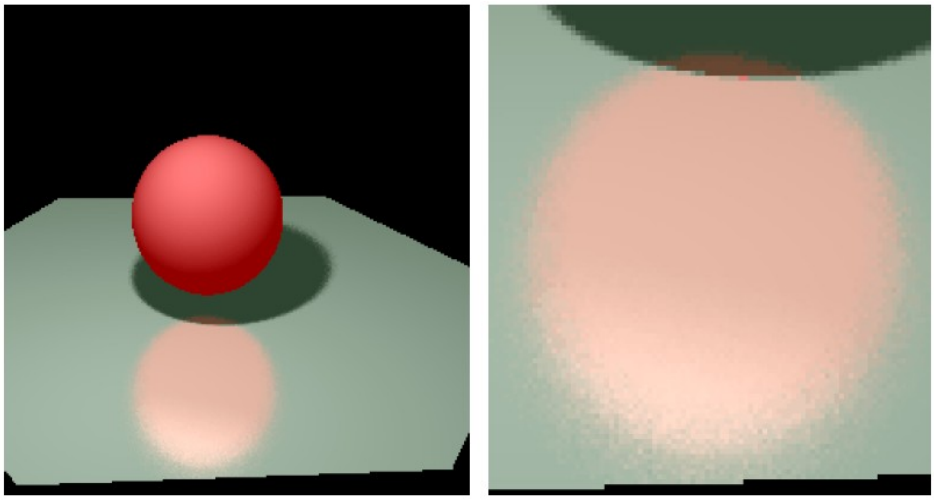
Lesklé povrchy můžeme vytvořit distribuováním odražených paprsků ve směru hlavního odrazu a následném zprůměrováním hodnot, které jsme těmito paprsky dostali. Výpočet distribuovaných odražených paprsků vypadá následovně:



Obr. 7.9. Ukázka odrazu – tradiční metoda. Převzato z [4]



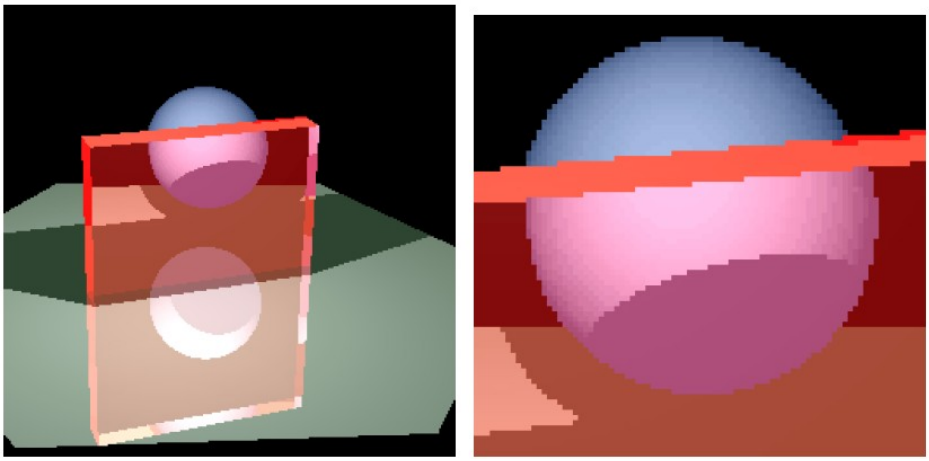
Obr. 7.10. Ukázka odrazu – 10 paprsků. Převzato z [4]



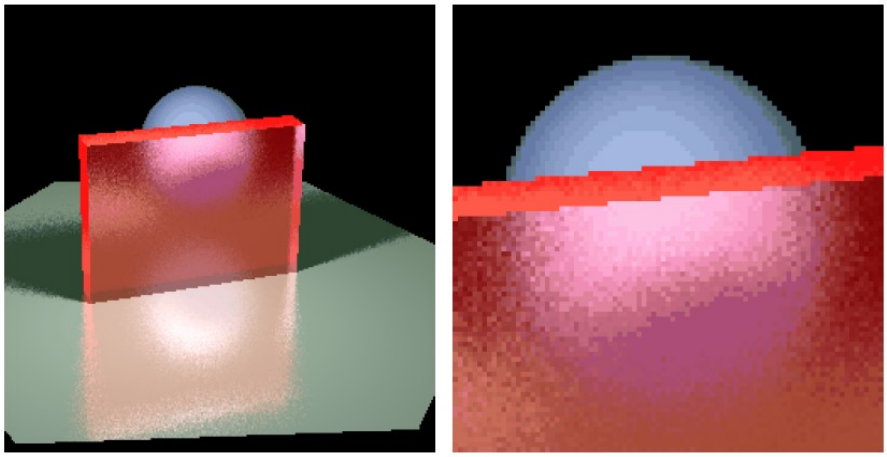
Obr. 7.11. Ukázka odrazu – 50 paprsků. Převzato z [4]

7.2.3 Průsvitnost

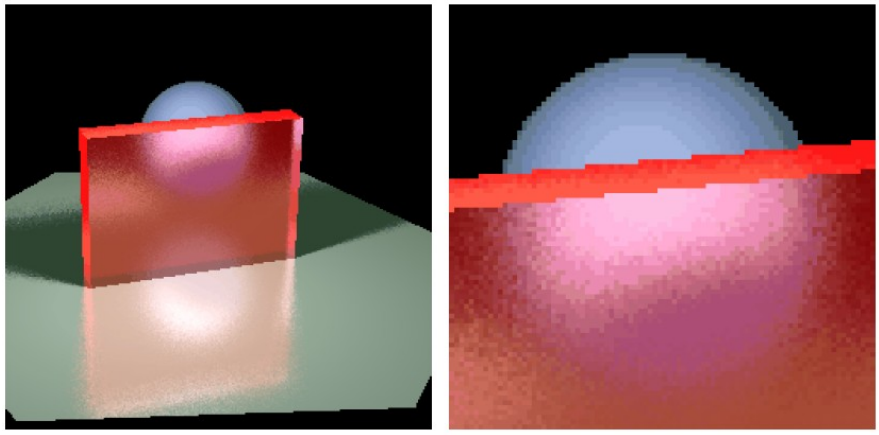
Tato kapitola byla převzata z [12]. Podobně jako u odrazu ani při lámání světla nedostaneme dokonalý obraz scény. Obraz, který je zobrazen přes průsvitné materiály bude také rozmazán. S touto vlastností si klasická metoda jen těžko poradí. Použijeme-li ale místo jednoho lomeného paprsku svazek paprsků různým způsobem distribuovaných, průměrná hodnota tohoto svazku nám tento efekt zajistí.



Obr. 7.12. Ukázka průsvitnosti – tradiční metoda. Převzato z [4]



Obr. 7.13. Ukázka průsvitnosti – 10 paprsků. Převzato z [4]

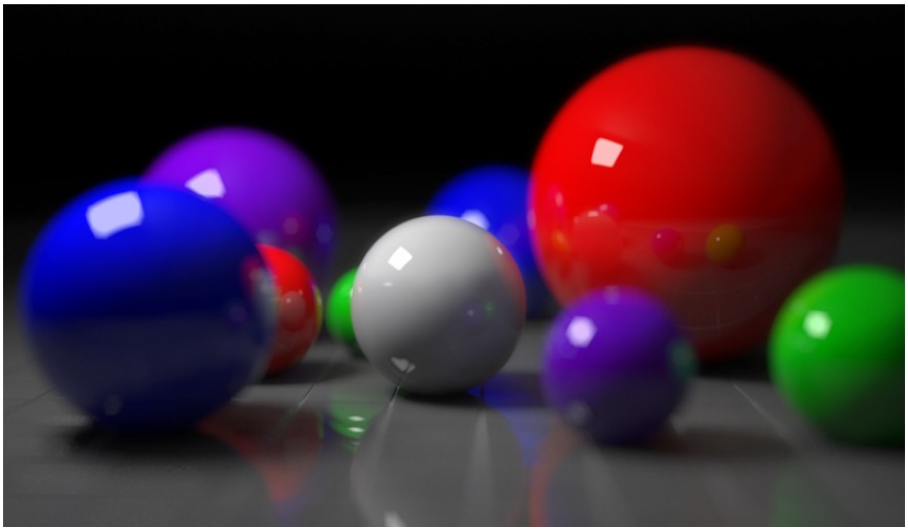


Obr. 7.14. Ukázka průsvitnosti – 20 paprsků. Převzato z [4]

7.2.4 Hloubka ostrosti

Tato kapitola byla převzata z [12]. Jak lidské oko tak i fotoaparáty mají omezenou clonu a tedy i omezenou hloubku ostrosti. Objekty, které jsou příliš daleko nebo příliš blízko jsou zobrazeny rozostřeně a rozmazaně. Téměř všechny techniky zobrazování v počítačové grafice používají model dírkové komory (angl. pinhole camera). V tomto modelu jsou všechny objekty bez ohledu na vzdálenost perfektně zaměřeny. V mnoha ohledech je to výhodné. Nicméně simulování hloubky ostrosti vede k realističtěji vypadajícím obrazům.

V distribuovaném sledování paprsku vytvoříme hloubku ostrosti tím, že umístíme čočku do přední části projekční roviny. Náhodně rozložené paprsky jsou opět použity pro simulaci rozmazání hloubky ostrosti. První vyslaný paprsek je čočkou nezměněn. Předpokládá se, že ohnisko čočky je v pevné vzdálenosti podél tohoto paprsku. Ostatní paprsky poslané stejným pixelem budou roztroušeny po povrchu čočky. Na bodech scény, které jsou blízko ohnisku čočky, bude ostrý obraz. Body nacházející se blíž nebo dál budou rozmazané.

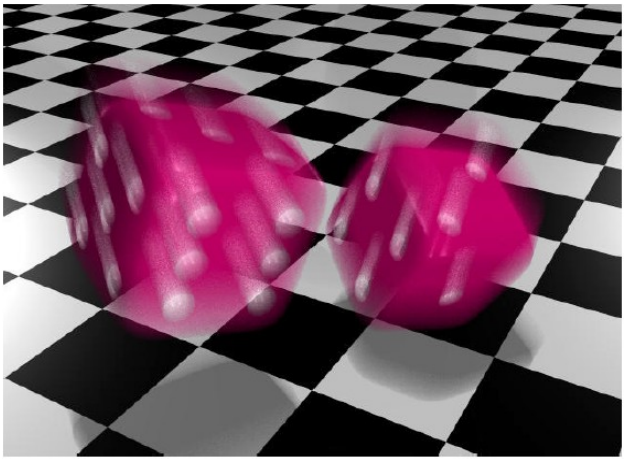


Obr. 7.15. Ukázka hloubky ostrosti. Převzato z [4]

7.2.5 Rozmazání pohybem

Tato kapitola byla převzata z [12]. Animace v počítačové grafice se vyrábí generováním statických obrázků a následným přehráním v daném pořadí. Jedná se o další vzorkovací proces, který je ovšem spíše prostorový než časový. Ve filmové kameře každý snímek představuje scénu v době, kdy byl snímek vytvořen. Jsou-li objekty scény v pohybu vzhledem k fotoaparátu, pak se objeví na snímku rozmazaně.

Metoda distribuovaného sledování paprsků může toto rozmazání simulovat pomocí svazku paprsků, který může být vyslán stejně dobře jak prostorově tak časově. Před každým vržením paprsků, jsou objekty umístěny do požadované pozice pro daný snímek. Aktuální hodnota paprsku je pak opět dána průměrem jednotlivých paprsků. Objekty pohybující se největší rychlostí budou tedy nejvíce rozmazány.



Obr. 7.16. Ukázka rozmazání pohybem. Převzato z [4]

8 Procedurální textury

Texturování je metoda, která nám umožňuje dodat větší realističnost 3D modelům uvnitř naší scény. Existuje několik druhů textur, které podle účelu dělíme na difúzní, normálové, odrazové, lomu, pohlcování světla a na textury prostředí [14]. Bavíme-li se o textuře, nejčastěji máme na mysli texturu difúzní. Difúzní textura nám popisuje, jak vypadá povrch našeho modelu při rovnoměrném nasvícení. S touto texturou si ve většině případech vystačíme. Chceme-li dosáhnout větší realističnosti, můžeme přidat normálovou texturu. Pomocí ní lze vytvořit dojem nehladkého povrchu. Tato textura se využívá při tzv. „bump mappingu“. V počítačové grafice se také dosti často používá textura prostředí, díky které lze navodit dojem kovového odlesku materiálu. Technika, která s ní pracuje se nazývá „mapování prostředí“ (angl. environment mapping). Ostatní typy textur jsou používány v menší míře.

Textury dělíme do dvou základních skupin. Máme textury rastrové a procedurální. Toto rozdělení je založeno na tom, jak jsou textury vytvářeny. Rastrové textury jsou reprezentovány předem vytvořeným rastrovým obrázkem. U tohoto typu textur je důležité, aby rastrový obrázek byl dostatečně kvalitní a detailní. Nevýhodou těchto textur je jejich konečnost a závislost na rozlišení. Také mohou vznikat různé problémy při nanášení textury na různé tvary objektů.

Tyto problémy nám odpadávají, pokud použijeme textury procedurální. Každá procedurální textura je tvořena posloupností matematických úkonů, které vedou k jejímu výslednému vzhledu. Takto vzniklé textury mají obrovské množství výhod. Mezi ty hlavní patří jejich nekonečnost a nezávislost na rozlišení. Jediný problém je, že ne všechny povrchy jdou matematicky popsat.

Procedurální textury můžeme mít 2D nebo 3D. Dále je pak rozdělujeme podle stylu vytváření na přímé, celulární a genetické.

8.1 Přímé textury

Tato kapitola byla převzata z [15]. Přímými texturami myslíme takové textury, které jsou vytvořeny pomocí dvojrozměrné nebo trojrozměrné funkce, která definuje barvu každého bodu v prostoru. V přírodě existuje hodně materiálů, jejichž vzhled lze matematicky popsat. K popisu některých nám stačí jednoduchá funkce, jiné musíme vytvořit pomocí složitých algoritmů. Spousta takovýchto textur využívá jako základní funkci Perlinův šum. Perlinův šum nám dovoluje vnést do textury požadovaný vzhled náhodnosti. Mezi procedurálními texturami jsou přímé textury nejrozšířenější.

8.2 Celulární textury

Tato kapitola byla převzata z [15]. Jedná se o textury, které jsou založeny na celulární funkci. Tato funkce je obdobně jako Perlinův šum použita jako výchozí funkce, nad kterou lze provádět další operace. Hlavní myšlenkou celulárních textur je vygenerování bodů v prostoru, které nám posléze slouží jako referenční body. Dále je vytvořena funkce určující vzdálenost pro daný bod v prostoru od nejbližšího z referenčních bodů. Podle této vzdálenosti je vyhodnocena barva pro daný bod. Touto metodou lze vytvořit textury jako je kůže, zmuchlaný papír, led, skála a jiné.

8.3 Genetické textury

Tato kapitola byla převzata z [15]. Do teď bylo na člověku, aby vytvořil texturu sám. Začal s výchozím algoritmem, který neustále vylepšoval, upravoval a testoval, až se nakonec víceméně přiblížil k požadovanému vzhledu. Pro genetickou tvorbu textur volíme trochu jiný přístup. Tvorba textur je postavena na genetickém algoritmu. Vygeneruje se soubor kandidátních textur. Pak nastoupí člověk k jejich estetickému ohodnocení a vybere několik z nich. Jejich mutací a křížením se vytvoří nové kandidátní textury. Tento postup se opakuje tak dlouho, dokud se nedosáhne požadovaného výsledku. Popis výsledné textury pak použijeme při texturování.

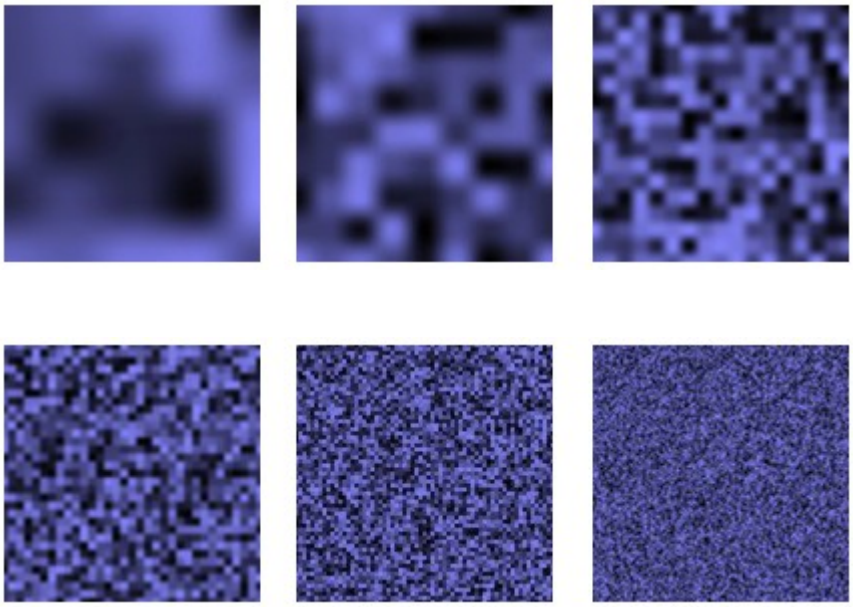
8.4 Perlinův šum

Tato kapitola byla převzata z [16]. Větší realističnost u textur získáme, zahrneme-li do jejich výpočtu funkci šumu. Na tuto funkci máme určité požadavky. Generátor, který používáme pro její vytvoření, nám musí vždy pro určitý bod v prostoru vracet totožné výsledky. Toto je základní předpoklad, aby se nám vždy vytvořila stejná textura. Textura, která by se po každém překreslení změnila, je pro nás ve většině případech nepoužitelná.

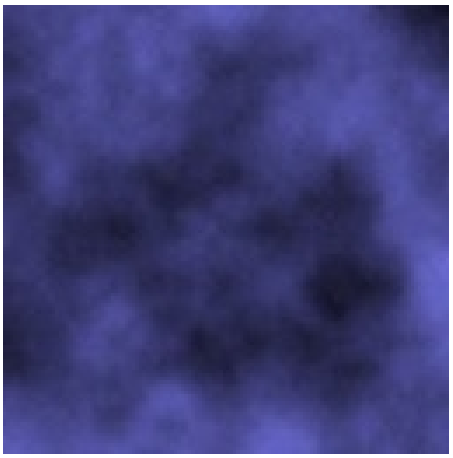
Máme-li funkci šumu, otázkou je, v jakém měřítku jej aplikovat. V přírodě si můžeme všimnout, že spousta věcí má fraktální charakter. Takové věci obsahují různé úrovně detailů. Vezmeme si třeba obrys pohoří. Vidíme na něm hory (velké rozdíly). Na každé hoře jsou patrné kopce (střední rozdíly). Na kopcích se vyskytují balvany (menší rozdíly) a kameny (drobné rozdíly).Takovéto vlastnosti, řekněme šum na několika úrovních, si lze všimnout všude. Dosti často jej nalézáme i ve struktuře materiálů a musíme jej tedy brát v úvahu při jeho popisu. K popisu takových materiálů využíváme jako základní funkci Perlinův šum.

Šum, který vyvinul Ken Perlin, je postaven na této základní myšlence. Jako stavební kameny používáme funkci šumu a funkci interpolace. Pro určité měřítko vytvoříme šum, který posléze

vyhladíme pomocí interpolace. To, pro jakou úroveň šum vytváříme, je dáno frekvencí, kterou jsme při vytváření šumu použili. Jak bude šum na dané úrovni vypadat, je ovlivněno amplitudou. Nakonec musíme jednotlivé úrovně šumu spojit do jednoho celku. Použitím Perlinova šumu dosáhneme daleko kvalitnějších výsledků než při použití šumu klasického.



Obr. 8.1. Vytvoření šumu pro různá měřítka. Převzato z [16]



Obr. 8.2. Spojení šumů různých měřítek a aplikování interpolace. Převzato z [16]

8.4.1 Funkce šumu

Funkce šumu je vlastně generátor pseudonáhodných čísel. Tento generátor má jako vstupní parametr celé číslo a na základě tohoto parametru vrací vygenerovanou hodnotu. Pokud spustíme generátor dvakrát se stejným vstupním parametrem, tak nám vždy vrátí stejný výsledek. Takovéto chování generátoru je velice důležité, jinak by funkce Perlinova šumu vracela nesmysly.

8.4.2 Funkce interpolace

Funkce interpolace spojuje body, které nám vrátila funkce šumu. Standardní funkce interpolace má tři parametry. Označme si tyto parametry jako A, B a C. A a B obsahují hodnoty, které chceme spojit. Parametr C je pak číslo od 0 do 1. Na tomto parametru závisí vrácená hodnota. Je-li roven 0, funkce interpolace vrací číslo A, je-li roven 1, pak vrací B. V jiném případě vrací nějakou hodnotu mezi A a B. Podle druhu funkce interpolace je dáno, jak budou v textuře zřetelné přechody mezi různými hodnotami.

9 Řešení problematiky

Cílem diplomové práce bylo vytvoření a zobrazení šachové scény v zadaném stavu. Před samotným vývojem bylo důležité se rozhodnout, jakým způsobem budou jednotlivé figurky reprezentovány. Jelikož se scéna měla zobrazovat metodou sledování paprsku, nejlepší možnou reprezentací na jednotlivé objekty uvnitř scény byla konstruktivní geometrie. Tato metoda byla nakonec i použita.

Následně byla vytvořena aplikace využívající k vykreslování scény klasické metody sledování paprsku, aby mohly být vyvíjeny a testovány základní tělesa. Tyto tělesa se posléze staly primitivy při tvorbě složitějších modelů figurek.

Když byla primitiva hotova, nastal čas pro tvorbu CSG stromu, abychom mohli vytvořit scénu. Bylo nutno definovat si jednotlivé operace a následně je implementovat. Jakmile byla konstruktivní geometrie v aplikaci implementována, mohlo se začít s tvorbou modelů figurek. Během tvorby figurek bylo zjištěno, že se samotnými primitivy nelze vystačit a musela být škála základních stavebních kamenů pro figurky rozšířena. Rozšířit se jí povedlo aplikováním různých transformací nad primitivy. S primitivy a s možností je transformovat již šli modely figurek vytvořit.

Po vytvoření scény nastal čas upravit klasickou metodu sledování paprsku na její distribuovanou podobu. Mohli jsme se tak zbavit aliasingu v obraze, získaly jsme jemné stíny a lesklé povrchy. Následovalo urychlování metody. Také byla snaha o vytvoření rychlejší metody pro tvorbu jemných stínů.

Poslední fáze vývoje byla věnována tvorbě procedurálních textur. Byla vytvořena funkce Perlinova šumu, jakožto základ algoritmů jednotlivých textur. Po té následoval vývoj textur samotných.

V následujících podkapitolách si přiblížíme, jakým způsobem byla tato problematika řešena.

9.1 Vytváření scény

Abychom mohli vytvořit a zobrazit nějakou scénu, musíme si nejprve uvědomit, z jakých stavebních kamenů je scéna poskládána. Žádná scéna se neobejde bez kamery. Nemáme-li kameru, pak nic nezobrazíme. Dále ve scéně existují zdroje světla. Pokud bychom neměli ve scéně světlo, pak bychom také nic neviděli. V neposlední řadě máme nějaké objekty, které chceme zobrazit. Tyto objekty musíme být schopni nějakým způsobem popsat. Čím jednodušší popis, tím lépe se nám s objekty pracuje. Proto jsou většinou objekty dekomponovány na menší snáze popsatelné části. Jak již bylo zmíněno v teoretické části, těm nejjednodušším objektům říkáme primitiva.

V následujících podkapitolách si uvedeme, jakým způsobem reprezentujeme kameru, světla a z kterých primitiv byla šachová scéna vytvořena.

9.1.1 Kamera

Pokud chceme získat obrázek z nějaké reálné scény za pomoci kamery, fotoaparátu nebo jiného zařízení, tak to jaký obrázek nakonec dostaneme, závisí na několika faktorech. Těmito faktory jsou pozice snímacího zařízení, jeho nasměrování a jeho zaostření. Tyto vlastnosti snímacího zařízení musíme namodelovat do kamery v naší scéně.

Jak již bylo popsáno v kapitole 3.1, paprsek je reprezentován touto rovnicí:

$$\mathbf{R=O+tD} \tag{3.1}$$

Paprsek se skládá ze dvou částí, jeho počátku **O** a určení směru **D**. Výchozí pozice nám bude dobře reprezentovat umístění kamery. S určením směru to není zase tak jednoduché. Obraz, který ze scény pořídíme, bude zobrazen v průmětně. Průmětna je popsána polem pixelů o dané šířce a výšce. V teoretické části jsme si uvedli, že paprsek vrháme do scény skrze jednotlivé pixely průmětny. Jak tedy určíme směr kamery, když každý paprsek směřuje jinam? Teoreticky lze říci, že směr je určen výchozí pozicí paprsku a umístěním středu průmětny. Aby však kamera fungovala správně, musíme mít na mysli jednu věc. Průmětna musí být kolmá na paprsek poslaný jejím středem. V jiném případě bychom ze scény dostávali nereálně vypadající obraz.

Posledním faktorem ovlivňujícím kameru je jeho zaostření. Zaostření modelujeme vzdáleností průmětny od počátku kamery.

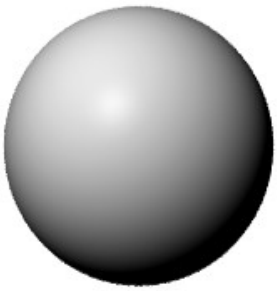
9.1.2 Zdroje světla

V klasické metodě sledování paprsku se všechna světla modelují pomocí bodových zdrojů. V distribuovaném sledování paprsku jsou body nahrazeny nějakým primitivem, nejčastěji koulí. Pokud bychom chtěli hledat průsečík se zdrojem světla, pak jej budeme hledat stejně jako s primitivem, které daný zdroj popisuje.

9.1.3 Jednoduché objekty scény

Jak jsme si popsali výše, objekty scény jsou popsány pomocí primitiv. Při řešení šachové scény byly použity jako základní stavební kameny následující primitiva: koule, válec, kvádr a hyperboloid. Z těchto primitiv je celá scéna postavena. Jakým způsobem se hledají průsečíky paprsku s matematicky popsaným objektem jsme si uvedli v kapitole 3.2. V této kapitole si ukážeme jak jsou daná primitiva popsána a kterými parametry lze modifikovat jejich vzhled. Následující vzorce platí pro tělesa umístěná v počátku souřadnic.

Koule:
$x^2+y^2+z^2 = r$

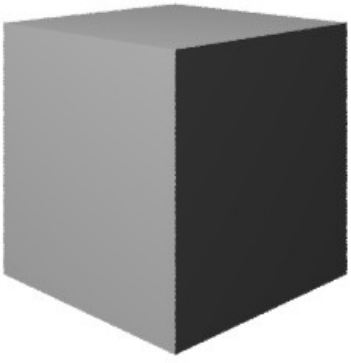


Válec s osou v y:
$x^2+z^2 = r$

Hyperboloid s osou v y:
$x^2-y^2+z^2 = r$

r - udává poloměr středu hyperboloidu

Kvádr je obecně dán minimálním a maximálním bodem v prostoru.



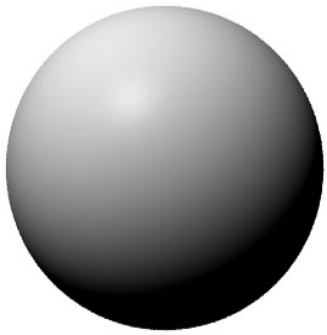
Obr. 9.1. Primitiva šachové scény

9.1.4 Transformace objektů

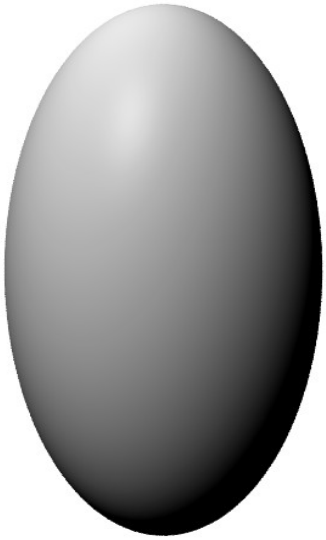
Jak probíhá transformace objektů u metody sledování paprsku je popsáno v kapitole 3.5.2. Stejný postup byl použit i při tvorbě šachové scény. Nejprve musíme vytvořit transformační matici, která nám popisuje, jakým způsobem chceme objekt transformovat. Mějme na paměti, že transformovaný objekt musí být umístěn v počátku souřadnic. Musíme tedy transformovaný objekt nejprve přemístit do počátku souřadnic, poté provedeme požadované transformace a posléze objekt zase vrátíme na původní místo. Také nesmíme zapomenout na to, že transformace jsou aplikované v opačném pořadí, musíme tedy postup zadávání příkazů při vytváření transformační matice invertovat.

Poté co máme vytvořenou transformační matici, tak už ji stačí jen přiřadit k nějakému objektu, popřípadě uzlu CSG stromu. Důležité je si pamatovat, že transformujeme paprsek a tedy pracujeme s inverzní transformační maticí. Jelikož vytvořená transformační matice bude stále stejná, je dobré si inverzní matici vypočítat pouze jednou a někam ji uložit.

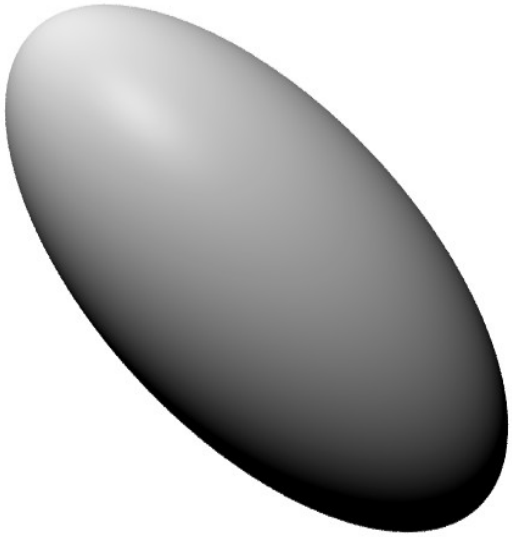
Při vytváření šachové scény jsme si vystačili pouze s transformací primitiv. Rutina transformace paprsku, která by normálně byla v CSG stromu u transformovaného uzlu, mohla být při průchodu CSG stromem vynechána. Průchod stromem je tak urychlen. Transformace paprsku je vložena k rutině hledání průsečíku s objektem. Výsledkem je, že paprsek je transformován až tehdy, kdy je to potřeba.



Obr. 9.2. Ukázka transformace – netransformované těleso



Obr. 9.3. Ukázka transformace – zvětšení v ose y



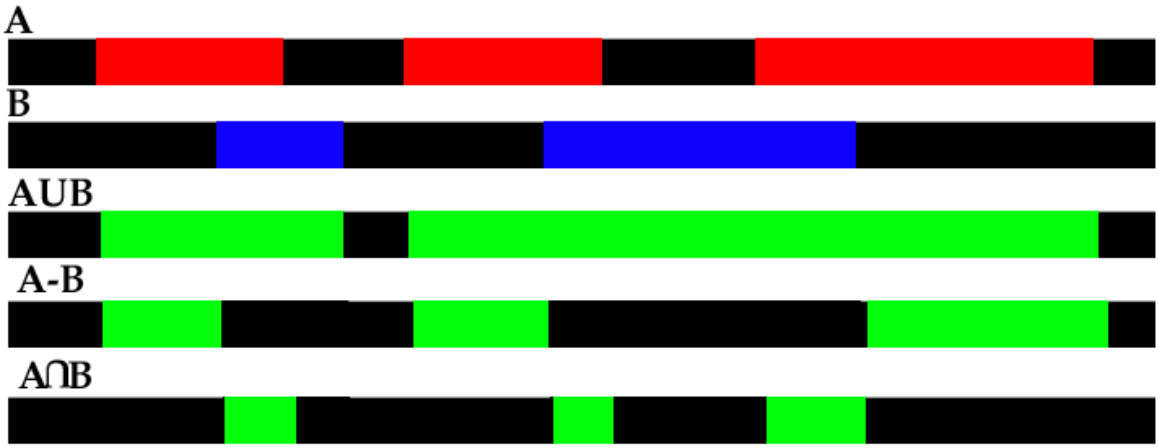
Obr. 9.4. Ukázka transformace – zvětšení v ose y a rotace podle osy z

9.1.5 Reprezentace složitých modelů

Pro tvorbu a reprezentaci modelů scény byla využita metoda konstruktivní geometrie. Tato reprezentace byla zvolena vzhledem k dobré spolupráci s metodou sledování paprsku. V teoretické části jsme si uvedli, jak prochází paprsek jednotlivými uzly CSG stromu a určuje průsečíky, které s nimi má. V této kapitole si popíšeme, jak byl CSG strom řešen při tvorbě šachové scény.

Víme, že objekt můžeme definovat pomocí intervalů na paprsku, jejichž hranice jsou dány jednotlivými průsečíky. Toho využijeme při tvorbě CSG stromu. U samotných primitiv nám stačilo vracet vždy nejbližší průsečík. U vytváření CSG stromu je ale musíme znát všechny, neboť dopředu nevíme, jaké operace budou prováděny. Jelikož chceme pracovat s intervaly popisující náš objekt, pak je důležité abychom vždy měli sudý počet průsečíků. Může se stát, že nalezneme jen jeden průsečík s primitivem, například pokud testujeme průsečík s rovinou. V takovém případě daný průsečík určuje oba konce intervalu, jinými slovy uložíme jej do seznamu dvakrát.

Každý uzel je tedy popsán seznamem hodnot, které nám udávají vzdálenosti na paprsku, kde jsme vstoupili nebo vystoupili z objektu. Máme tedy dva seznamy, jeden od levého a druhý od pravého synovského uzlu. Z těchto dvou seznamů vytvoříme jeden výsledný. Jak to provedeme záleží na požadované operaci.



Obr. 9.5. Operace nad seznamy v CSG stromu. Každé dvě hodnoty v jednotlivých seznamech reprezentují část paprsku uvnitř objektu

Průchod CSG stromu řešíme rekurzivně. U kořene dostaneme seznam platných průsečíků. Z tohoto seznamu vybereme nejmenší kladnou hodnotu a tu vracíme jako nalezený průsečík komplexního objektu.

Nesmíme však opomenout, že vracení samotného průsečíku nám nestačí. Díky průsečíku sice známe bod, kde jsme na komplexní objekt narazili, avšak kvůli stínování, a nejen kvůli němu, musíme znát i směr normály v tomto bodě. Máme dvě možnosti jak toto řešit. Můžeme si vypočítat normálu u každého průsečíku a pak si ji pamatovat nebo si pamatovat primitivum, ke kterému průsečík patří a poté, až bude potřeba, si normálu dopočítat. Jelikož počítat normálu u každého průsečíku je zbytečné, neboť se nakonec použije jen jedna, byla zvolena při vytváření šachové scény druhá možnost.

Obr. 9.6. Vytvoření komplexního objektu pomocí CSG stromu

9.2 Vykreslení scény

Pro vykreslení scény je implementována metoda distribuovaného sledování paprsku. Tato metoda je zvolena vzhledem k realističtějšímu zobrazení oproti klasickému přístupu. Získáme tedy kvalitněji vypadající scénu, ovšem je třeba si uvědomit, že výpočetní doba se nám velmi prodlouží.

Takto vypadá pseudokód pro distribuované sledování paprsku:

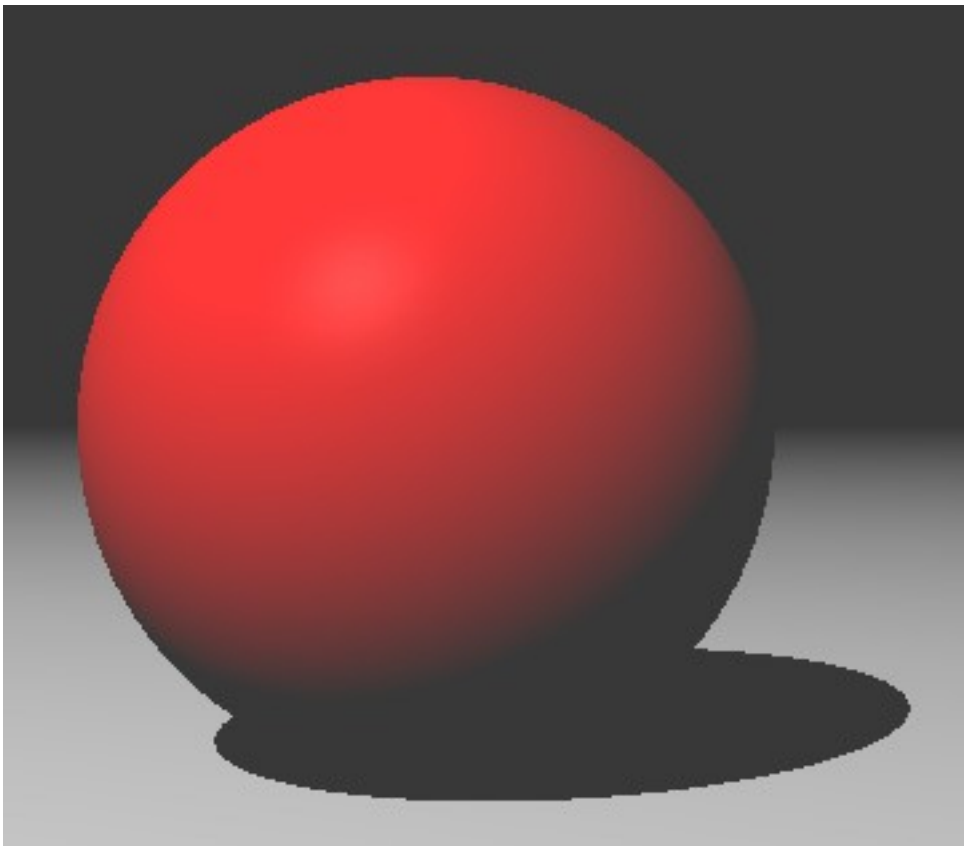
```
...
Pro každý pixel průmětny:
        pro každý primární paprsek
                distribuuj bod uvnitř pixelu
                bodem pošli paprsek a proveď rutinu pro sledování paprsku
        vypočítej výslednou barvu pixelu
        obarvi pixel touto barvou
...
```

```
...
Rutina pro paprsek:
        nalezni nejbližší objekt a urči průsečík s tímto objektem
        pro každé světlo:
                distribuuj stínové paprsky
                urči stín od každého paprsku
                vypočítej výslednou hodnotu stínu
        proveď výpočet difúzní složky
        proveď výpočet lesklé složky
        pro každý distribuovaný odražený paprsek:
                nalezni jeho směr
                tímto směrem vrhni paprsek a proveď rutinu pro sledování paprsku
        vyhodnoť výslednou hodnotu, kterou vrací svazek odražených paprsků
        pomocí osvětlovacího modelu urči barvu, kterou bude rutina vracet
...
```

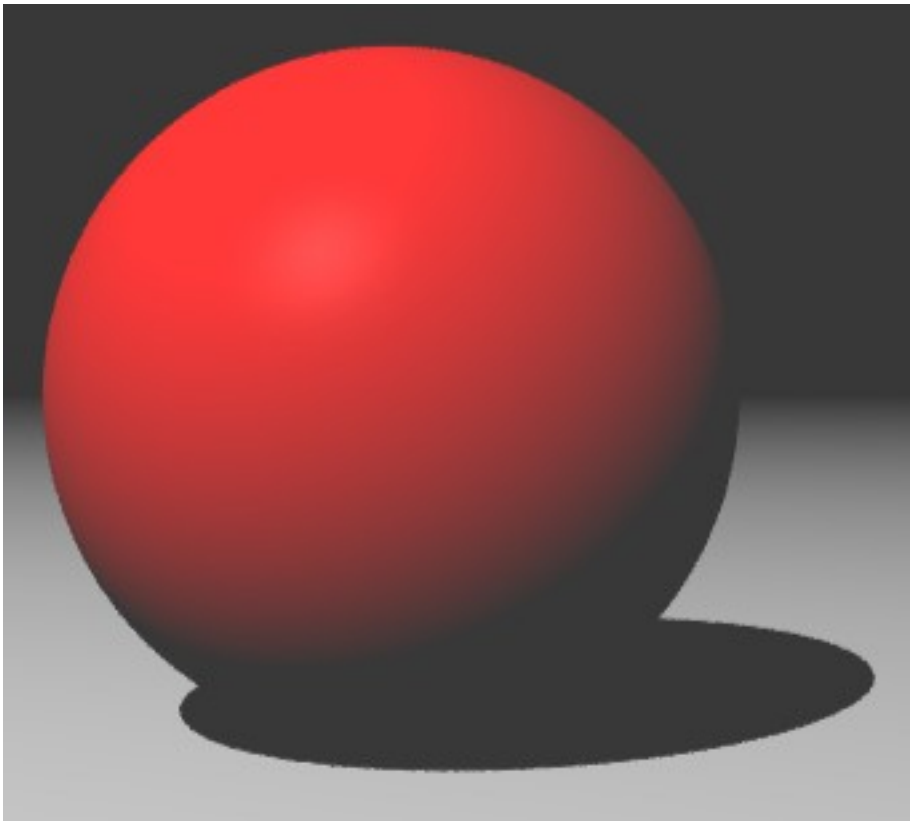
9.2.1 Distribuované sledování primárních paprsků

K čemu je dobré distribuované sledování primárních paprsků jsme si popsali v teoretické části. Nyní se budeme zabývat, jakým způsobem byla distribuce primárních paprsků řešena při vykreslování šachové scény. Bereme v úvahu, že máme svazek n primárních paprsků. Tyto paprsky chceme nějakým způsobem distribuovat. Z důvodů uvedených v teoretické části chceme, aby tyto paprsky byly pokud možno v každém pixelu distribuovány jiným způsobem. Nejlepším řešením bude náhodně vygenerovat v každém pixelu n bodů, které nám určí směry jednotlivých paprsků.

Distribuované sledování primárních paprsků bylo pojato, jako ne zcela nová, ale jako upřesňující metoda klasického přístupu. Směr prvního paprsku je zachován, posíláme ho tedy středem pixelu. Ostatní paprsky jsou náhodně v pixelu distribuovány. Má to tu výhodu, že i při malém počtu paprsků se nám nemůže stát, že bychom příliš malé objekty, které jsme klasickou metodou nalezli, minuly. Jde tedy pouze o upřesnění barvy pixelu vzhledem k jeho okolí.



Obr. 9.7. Klasická metoda sledování paprsku



Obr. 9.8. Distribuované sledování primárních paprsků – 9 paprsků

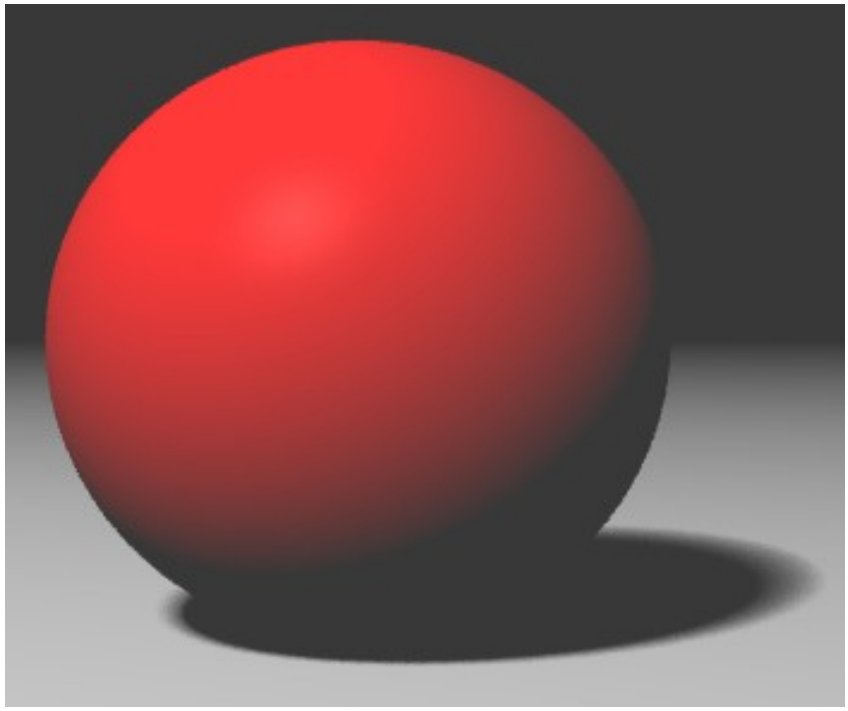
9.2.2 Distribuované sledování stínových paprsků

Distribuované sledování stínových paprsků přímo souvisí s tvorbou jemných stínů. Má tu výhodu, že rutina spojená se stínovým paprskem zůstává, pouze je provedena víckrát a v odlišných směrech. Nevýhodou je, že musíme poslat mnoho stínových paprsků, abychom dosáhli kvalitního výsledku.

V klasické metodě jsme poslali paprsek ke světelnému zdroji a testovali, zda existuje nějaký objekt, který mu cestu zahradí. Směr, kterým jsme paprsek posílali byl dán bodem, reprezentujícím světelný zdroj. V metodě distribuovaného sledování ovšem nevyužíváme bodové zdroje světla. Který bod tedy použít jako cíl stínového paprsku? Lze říci, že to bude jakýkoliv bod na povrchu tělesa znázorňující světelný zdroj.

K vytvoření jemného stínu využijeme metodu Monte Carlo. Náhodně distribuujeme body na povrchu světelného zdroje a k nim vyšleme stínové paprsky. Podle počtu prošlých a neprošlých paprsků určíme, jakou hodnotu bude stín mít.

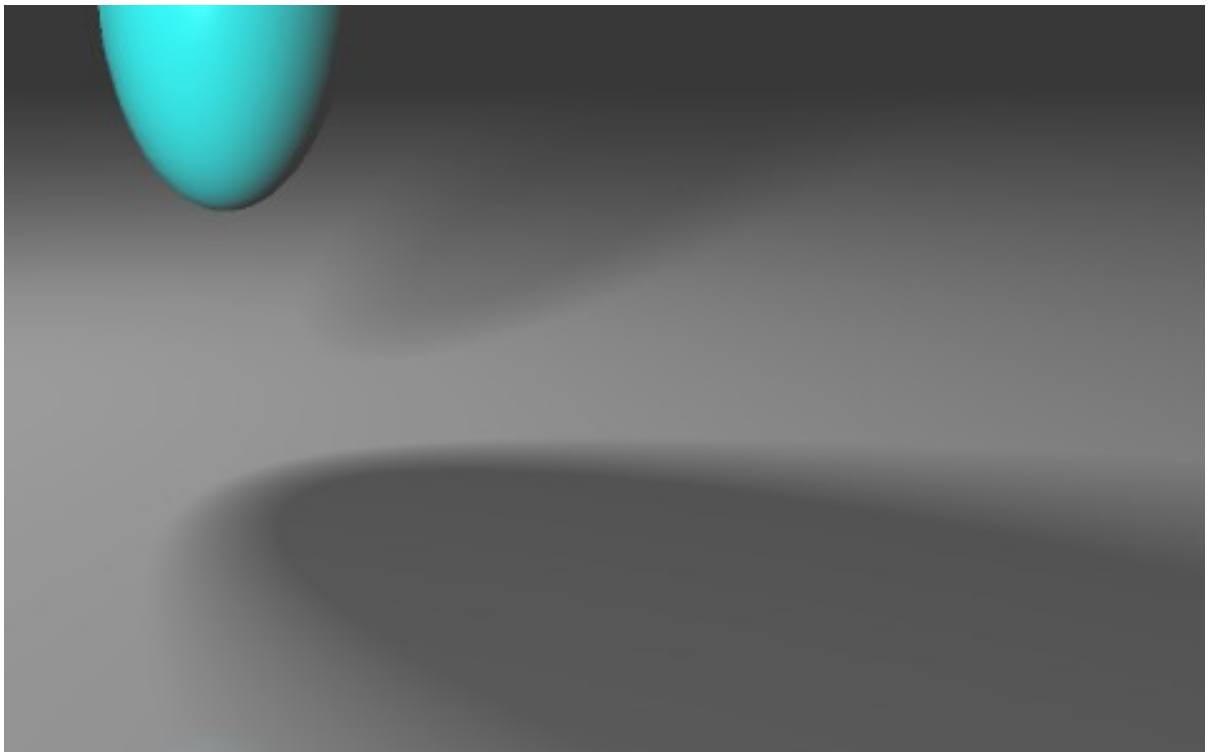
V šachové scéně jsou použity koule jako tělesa reprezentující zdroje světla. Známe-li střed koule, jak distribuovat n bodů po jejím povrchu? Jelikož víme, jaký je průměr koule, můžeme náhodně vygenerovat 3 hodnoty, které jsou menší. Tyto hodnoty nám můžou reprezentovat bod uvnitř čtverce, který kouli obklopuje. Vyšleme- li paprsek ze středu koule směrem k tomuto bodu, tak ve vzdálenosti rovnající se poloměru jsme dosáhli jejího povrchu. Takto můžeme nalézt všechny požadované body.



Obr. 9.9. Distribuované sledování stínových paprsků – 100 paprsků

9.2.3 Jemné stíny – jiný přístup

Jelikož k dosažení kvalitního jemného stínu je u distribuovaného sledování zapotřebí velkého množství stínových paprsků, byla snaha o použití jiné rychlejší metody. Velmi slibnou se jevila metoda „Měkkého stínu jedním vzorkem“, která je popsána v teoretické části. Použití teto metody algoritmus velmi rychlilo a výsledný stín byl velmi kvalitní. Bohužel použití této metody u jiných těles než je koule není triviální. Navíc vznikl problém jak tuto metodu zkombinovat s CSG stromem. Jak dopočítat stín u komplexních objektů vytvořených sadou množinových operací? Tato problematika vyžadovala velké množství času k jejímu prozkoumání. Jelikož výsledek nebyl zaručen, bylo od použití této metody nakonec upuštěno.



Obr. 9.10. Měkké stíny za pomoci metody s jedním vzorkem

9.2.4 Distribuované sledování odražených paprsků

Použití distribuovaného sledování odražených paprsků nám umožní vytvářet lesklé povrchy. Hlavní myšlenkou je distribuovat paprsky kolem vektoru určující směr odražení. Jaký rozptyl bude svazek mít je dáno materiálem povrchu, od kterého se paprsky odrážejí. Jak určit směr jednotlivých paprsků?

Distribuce odražených paprsků je řešena následovně. Mějme jednotkový vektor určující hlavní směr odražení paprsku. Můžeme si představit, že na konci tohoto vektoru máme kouli, jejíž poloměr je určen rozptylem materiálu. Potom můžeme distribuovat body na povrchu této koule, které nám

budu určovat směr jednotlivých odražených paprsků. Tím docílíme distribuci kolem hlavního vektoru o určitém rozptylu.

Obr. 9.11. Lesk zobrazený klasickou metodou

Obr. 9.12. Lesk zobrazen distribuovaným sledováním odražených paprsků – 5 paprsků

9.2.5 Osvětlovací model

Když máme výsledky od jednotlivých sekundárních paprsků, zbývá nám ještě určit, jak z nich vypočítat výslednou barvu, kterou jsme hledali primárním paprskem. K tomu slouží osvětlovací model. V teoretické části jsme si popsali několik osvětlovacích modelů. Vzhledem k tomu, že budeme chtít zobrazovat celou šachovnici, pak jednotlivé figurky budou natolik malé, že rozdíly mezi jednotlivými osvětlovacími modely, které máme k dispozici budou nepatrné. Nepotřebujeme tedy

extrémně složitý osvětlovací model a vystačíme si se základním osvětlovacím modelem od B.T.Phonga.

9.2.6 Urychlování vykreslování

Musíme mít na paměti, že při použitím distribuovaného sledování paprsku se každá část, u které jsme použili distribuci n paprsků, počítá n krát déle. Jelikož se jedná o rekurzivní algoritmus, pak přidáním více paprsků na určité místo se algoritmus mnohonásobně zpomalí. Nelze tedy použít naivní přístup bez optimalizací.

V teoretické části jsme si uvedli několik možností, jak sledování paprsku urychlit. Některé jsou pouze pozitivní a nevyplatí se je použít jen ve velmi malém procentu případů. Jiné s sebou nesou jisté komplikace. Optimalizace, které jsou prováděny během vykreslování šachové scény jsou zvoleny vzhledem k jejich kladům a záporům.

Byla snaha o to, aby všechny výpočty, které nemusejí být prováděni během vykreslování, byly předpočítány. Jelikož jsou použity transformace jen na primární tělesa, pak řešení transformace paprsku je převedeno z uzlu CSG stromu do rutiny hledání průsečíku s tímto tělesem. Vyhneme se tak zbytečnému ověřování zda a jak paprsek transformovat u případů, kde to není potřeba. Jednotlivé figurky mají své obalové těleso. Tím se nám sníží počet testovaných objektů. Pamatujeme si objekt, který jsme nalezli u vedlejšího pixelu. Hledáme tedy nejprve průsečík s tímto objektem a teprve potom s ostatními. Jelikož se hledá průsečík vždy do určité vzdálenosti na paprsku, spousta ostatních objektů neprojde tímto filtrem už na úrovní obalovacího tělesa. Ušetříme si tak zbytečné procházení CSG stromu u objektů, které by nakonec stejně nevyhovovaly.

Existují také optimalizace, které se nám v šachové scéně nevyplatí použít. Jednou takovou optimalizací je dělení prostoru. Uvažujeme-li klasickou šachovou scénu, pak figurky jsou relativně blízko u sebe, scéna je poměrně malá a na úrovni obalovacích těles nemáme příliš moc objektů k testování. Použití dělení prostoru v takové scéně by algoritmus nezrychlilo, ba naopak ještě více zpomalilo.

U distribuovaného sledování je jednou z oblastí, kde lze algoritmus výrazně zrychlit, tvorba jemných stínů. Byla snaha o použití metody „Měkkého stínu jedním vzorkem“. Tato metoda je velmi slibná, bohužel z výše uvedených důvodů bylo od ní upuštěno. Přesto i tato oblast byla urychlena. K tvorbě jemných stínů potřebujeme velké množství stínových paprsků. Tyto paprsky algoritmus mnohonásobně zpomalují a při tom v oblastech beze stínu jsou poslány naprosto zbytečně. Nemá tedy smysl vrhat vždy n paprsků, které potřebujeme k dosažení kvalitního jemného stínu. Můžeme testovat pár paprsků a pokud všechny tyto paprsky dorazí až ke světlu, pak můžeme usuzovat, že i ostatní by se dostaly ke zdroji světla a vyhodnotit výsledek tak, že se v oblasti stín nenachází.

Tento závěr si ale můžeme dovolit jen v případě, že vyhodnocujeme oblast jako beze stínu. V opačném případě to ale nelze. Musíme si uvědomit, že kdybychom nejprve poslali pár paprsků jako testovacích a určili podle nich oblast se stínem, polostínem a beze stínu, tak dále bychom zjemňovali jen oblast polostínu a hranice mezi stínem a polostínem by zůstala definovanou malým počtem paprsků a stín by byl nekvalitní. V oblasti polostínu by byly patrné některé pixely, u kterých při testování bylo vyhodnoceno, že patří do oblasti plného stínu. Ono se to v určité míře děje i při vyhodnocení, že se pixel v oblasti stínu nenachází. Ovšem stín už je v této oblasti natolik sláb, že to nejde skoro poznat a můžeme si to dovolit. Navíc tak dosáhneme velké zvýšení rychlosti na úkor malé ztráty kvality.

9.3 Procedurální textury

Aby naše modely ve scéně vypadaly více realističtěji, musíme jim přiřadit vzhled nějakého materiálu. Toho docílíme pomocí textur. Jak jsme si popsali v teoretické části, máme na výběr mezi texturami rastrovými a procedurálními. Jelikož v metodě sledování paprsku vždycky známe přesnou pozici bodu, u kterého budeme chtít znát jeho barvu v textuře, tak se nám bude daleko snáz pracovat s texturami procedurálními. Rastrové textury mají jednu hlavní nevýhodu. Tyto textury jsou vždy 2D a musíme je nějakým způsobem na objekt namapovat. Vzhled objektu bude tak vždy závislý na použité metodě mapování. Pokud použijeme 3D procedurální textury, pak nám tyto problémy odpadávají.

Během vývoje šachové scény bylo vytvořeno několik procedurálních textur. Většina z nich využívá jako základní funkci Perlinův šum s různými parametry. V následujících podkapitolách si uvedeme, jak byla funkce Perlinova šumu řešena a jakých výsledků jsme s ní dosáhli.

9.3.1 Perlin Noise

Existuje spousta zdrojů popisujících tvorbu Perlinova šumu. Problém je, že většina zdrojů uvádí 1D nebo 2D verzi. V šachové scéně potřebujeme Perlinův šum pro 3D. Jak jej vytvoříme? Pokud známe 1D a 2D variantu, lze podle vztahů, které mezi sebou mají, rozšířit 2D variantu na 3D. Takovýmto postupem byl vytvořen 3D Perlinův šum, který potřebujeme.

Nejprve musíme rozšířit generátor pseudonáhodných čísel. Používáme-li takovýto generátor pro 2D:

```
…
n=x+y*59;
n^=(n<<13);
výsledek=(1.0-(((n*(n*n*15731+789221)+1376312589)& 0x7fffffff)/1073741824.0));
…
```

tak jej jednoduše rozšíříme do 3D, nahradíme-li

```
n=x+y*59;
```
(9.1)

za

```
n=x+y*59+z*157;
```
(9.2)

Dobrých vlastností dosáhneme, pokud konstanta, kterou násobíme z-ovou souřadnici, bude prvočíslo. Také si všimněme, že každé přidání nové dimenze je násobeno konstantou o řád vyšší.

Když máme upraven generátor, musíme dále upravit funkci rozmazání. Funkce rozmazání se využívá k tomu, aby aktuální hodnota, s kterou dále pracujeme, brala určitou měrou v potaz svoje okolí. Hlavní myšlenkou je, rozhodnout se, které okolní body mají vliv na aktuální hodnotu a jakou měrou přispívají k jejímu vyhodnocení. Na obrázku 9.13 vidíme, jaké body jsou uvažovány v 2D verzi.

R S R
S C S
R S R

Obr. 9.13. Rozložení bodů ve 2D: R – roh, S – stěna, C – střed

Chceme, aby střed měl při výpočtu největší zastoupení, jednotlivé strany poloviční jak střed a rohy poloviční jak strany. Jak určitou procentuální zastoupení jednotlivých bodů nebo lépe řečeno, jak určit hodnoty, kterými budeme tyto body dělit, abychom tohoto procentuálního zastoupení dosáhli? Řešení je následující. Budeme postupovat od složek, které přispívají nejmenší měrou, tedy od rohů. Každému bodu náležícímu do určité skupiny přiřadíme hodnotu, která je dvojnásobná jak hodnota níže postavené skupiny. Začínáme u hodnoty 1. Ohodnocení bodů vidíme na obrázku 9.14.

1 2 1
2 4 2
1 2 1

Obr. 9.14. Ohodnocení jednotlivých bodů

Nyní spočteme sumu všech hodnot:

$$4*1 + 4*2 + 1*4=16 \tag{9.3}$$

A určíme dělitele výsledků z daných bodů:

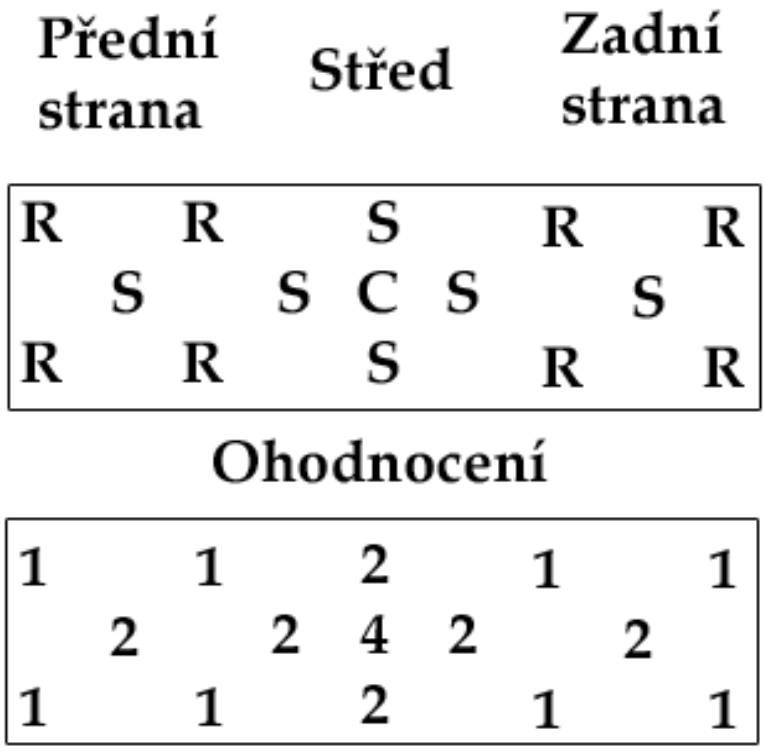
$$\text{střed}=16 : 4 \tag{9.4}$$

$$\text{strana}=16 : 2 \tag{9.5}$$

$$\text{roh}=16 : 1 \tag{9.6}$$

Pokud budeme jednotlivé body dělit danými hodnotami, dosáhneme takovéhoto procentuálního ohodnocení ke každému bodu: střed – 25%, strana – 12,5%, roh – 6,25%. Důležité je, aby součet nám dával hodnotu 1.

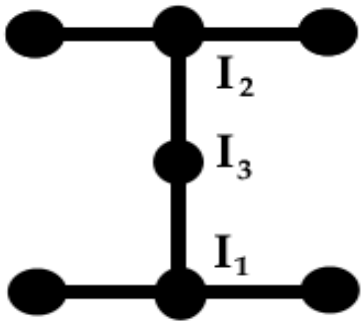
Ze znalostí tohoto postupu ohodnocování si můžeme vytvářet různé vlastní šablony pro funkci rozmazání. Šablonu pro 3D, která je použita v 3D verzi Perlinova šumu, vidíme na následujícím obrázku.



Obr. 9.15. Vytvoření šablony pro 3D

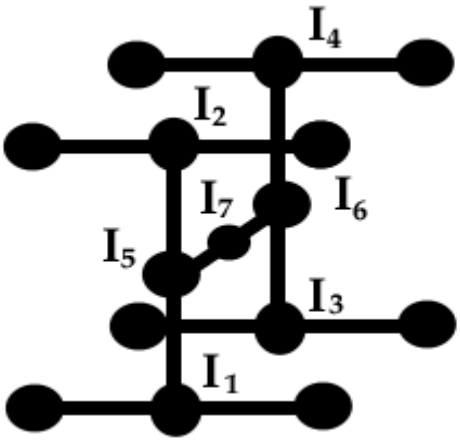
Použitím této šablony ohodnotíme body následovně: střed – 16,6%, strana – 8,33%, roh – 4,16%.

Ještě nám zbývá upravit funkci interpolace. Jak budeme interpolovat body v 3D? 2D variantu lze popsat obrázkem 9.16, kde Ix znázorňuje interpolaci mezi dvěma sousedními body. Bod, který zjemňujeme se nachází v levém dolním rohu. Podiváme se, jaké hodnoty mají vedlejší body a ty mezi sebou interpolujeme.



Obr. 9.16. Interpolace ve 2D

Na obrázku 9.17. je znázorněno, jak bude funkce interpolace vypadat v 3D. Bod, který zjemňujeme se opět nachází v levém dolním rohu.



Obr. 9.17. Interpolace v 3D

Nyní nám již nic nebrání v tom použít Perlinův šum v 3D.

9.3.2 Použité textury

V této kapitole si ukážeme některé textury, kterých bylo během vývoje dosaženo. Jak bude vidět, spousta textur vychází z výše zmiňované funkce Perlinova šumu.

Dřevo:

…

barva(0.65, 0.4, 0.15)

PN = 0.5 – PerlinůvŠum(2*x, 10*y, 2*z)

PN = cos(PN)*PN

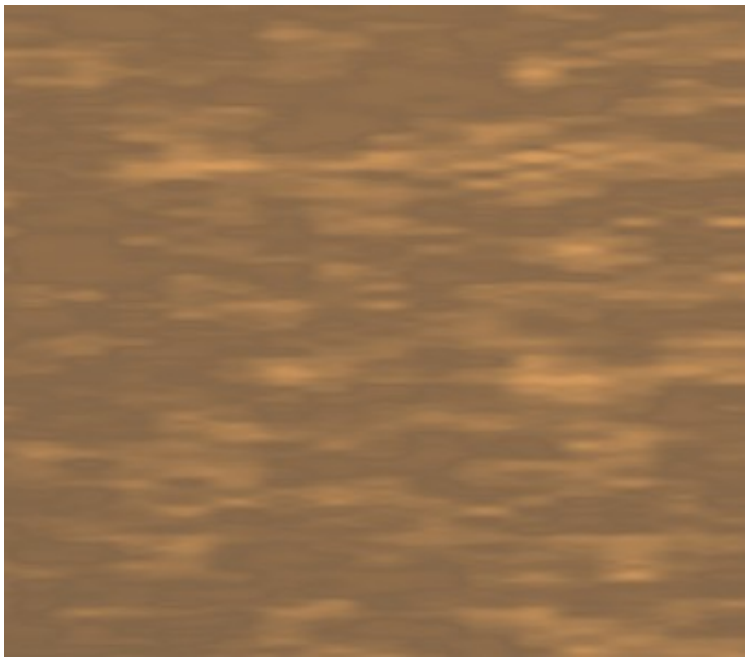
pokud je PN < 0, pak PN = 0

pokud je PN < 0.5, pak PN = 1 - PN

pokud je PN >1, pak PN = 1

výsledek = barva*PN

…



Obr. 9.18. Dřevo

Kámen:

…

barva(0.8, 0.8, 0.1)

PN = 0.5 – PerlinůvŠum(20*x, 100*y, 100*z)

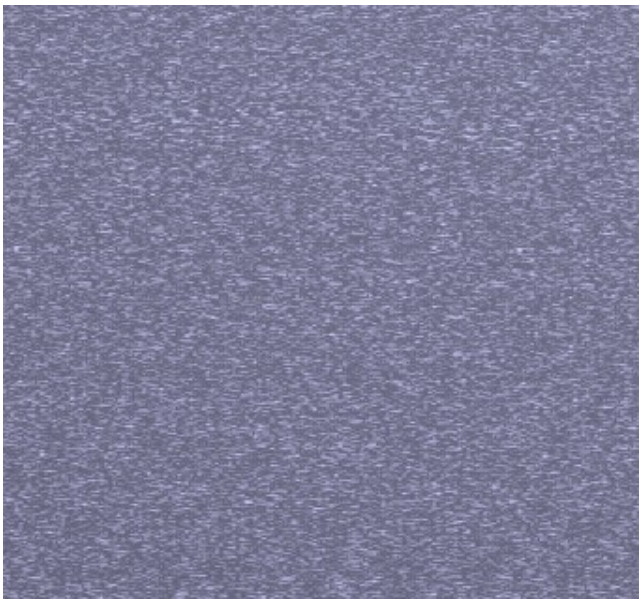
pokud je PN < 0, pak PN = 0

pokud je PN < 0.5, pak PN = 1 - PN

pokud je PN >1, pak PN = 1

výsledek = barva*PN

…



Obr. 9.19. Kámen

Mramor:

…

barva(0.8, 0.8, 0.1)

PN = 0.5 – PerlinůvŠum(3*x, 3*y, 3*z)

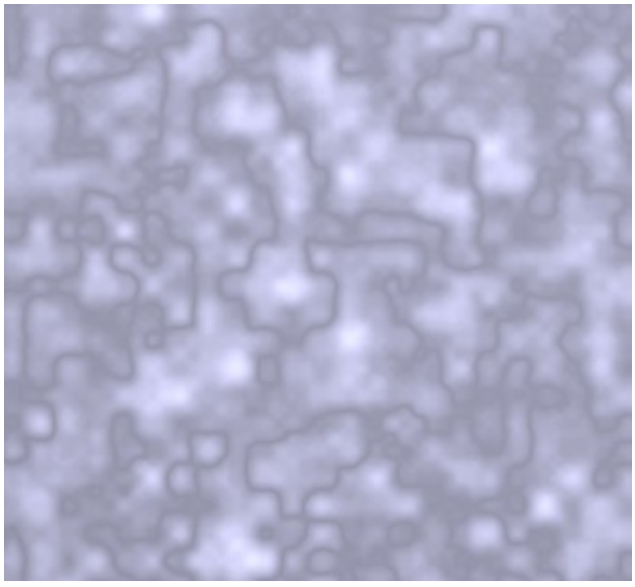
pokud je PN < 0, pak PN = 0

pokud je PN < 0.5, pak PN = 1 - PN

pokud je PN >1, pak PN = 1

výsledek = barva*PN

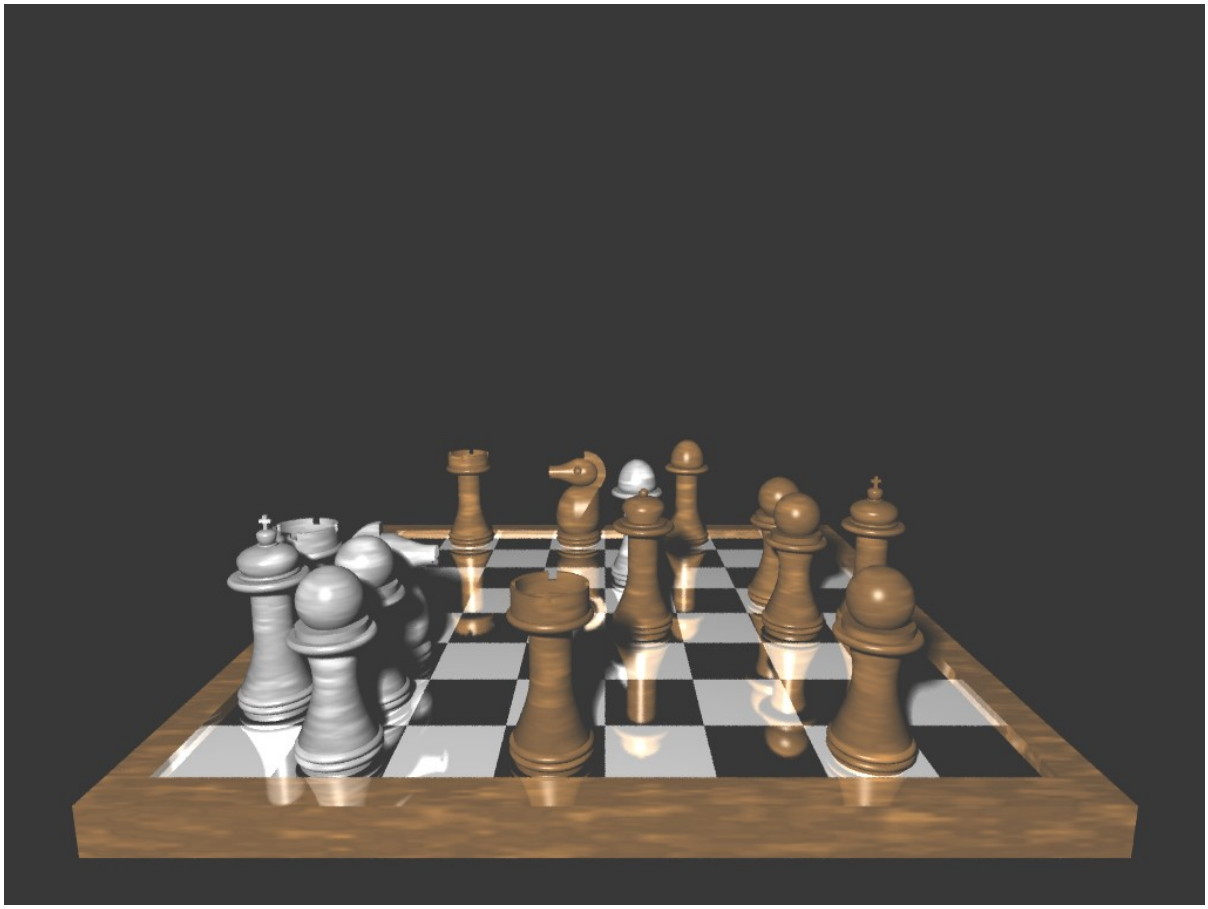
…



Obr. 9.20. Mramor

9.4 Zhodnocení

Byla vytvořena aplikace, která zobrazuje šachovou scénu. Rozestavení figurek si může uživatel definovat sám. Scéna je vykreslena pomocí metody distribuovaného sledování paprsku. Na jednotlivé figurky jsou aplikovány procedurální textury. Ve výsledném obrázku jsou patrné některé efekty vyplývající z použité metody, jako jsou měkké stíny, lesklý povrch atd. Ve výsledku tedy získáme daleko realističtěji vypadající scénu, než kdybychom použili pouze klasickou metodu.



Obr. 9.21. Výsledná scéna

Tohoto výsledku ale dosáhneme na úkor rychlosti vykreslování. Musíme si uvědomit, že v každém pixelu se provádí několikrát rutina spojená s rekurzivním sledováním paprsku. Musí se určit, kterou figurku paprsek zasáhne nejdříve, musíme procházet CSG stromem, abychom provedli operace nad jednotlivými primitivy a určili tak výsledný tvar figurky. Poté se musí pro daný bod na figurce určit jeho barva, která je dána texturou. Tedy musíme provést výpočet Perlinova šumu v 3D a provést s výsledkem jisté operace. Barva tohoto bodu je dále ovlivněna sekundárními paprsky.

Tohle všechno se pro každý pixel provádí několikrát. A to jsme doposud nebrali v úvahu distribuci primárních ani sekundárních paprsků, s kterými se to všechno ještě prodlužuje. Jelikož se

jedná o věci, které se provést musí, nelze je dost dobře optimalizovat. Jedinou optimalizací, která by výrazně algoritmus urychlila, je paralelní zpracování na více procesorech.

Chceme-li tedy dosáhnout kvalitního obrázku, musíme počítat s delší dobou vykreslování. Vykreslení šachovnice v počátečním stavu na rozlišení 800*600 procesorem AMD Athlon 2400+ trvá klasickou metodou 14 vteřin. Jak dlouho se bude scéna vykreslovat závisí na jejím rozestavění, rozlišení a počtu jednotlivých paprsků. Při kvalitním vykreslení scény ve vysokém rozlišení se můžeme dostat i do řádově hodin. Z těchto důvodů se metoda distribuovaného sledování paprsku nevyužívá pro aplikace reálného času a její uplatnění je v oblasti, kde na času vykreslování zase tak nezáleží.

10 Závěr

Výsledkem diplomové práce je aplikace vykreslující šachovou scénu. V této aplikaci lze zvolit rozmístění figurek dle vlastního výběru. Dále si uživatel může nastavit rozlišení, v kterém bude chtít vykreslovat, počet primárních a sekundárních paprsků. Může si tak sám zvolit kvalitu vykreslování. Vykreslování scény probíhá pomoci metody distribuovaného sledování paprsku. Lze tak dosáhnout efektů, které bychom klasickou metodou získat nemohli.

Byla snaha o použití metody „Měkkého stínu jedním vzorkem“ popsané v kapitole 7.2.1. S touto metodou bylo dosaženo velmi kvalitních jemných stínů. Bohužel díky problematickému nasazení této metody s operacemi v CSG stromě pro různá tělesa bylo nakonec od této metody upuštěno.

Dále byla implementována metoda konstruktivní geometrie, jakožto nástroj pro tvorbu šachových figurek. Figurky jsou tedy reprezentovány pomocí CSG stromu, v jehož listech se nacházejí primitiva. K rozšíření škály základních objektů byly implementovány transformace nad jednotlivými objekty.

Na objekty scény jsou použity procedurální textury. Byla implementována 2D funkce Perlinova šumu jako základní funkce pro tvorbu jednotlivých textur. Následně podle znalosti vztahů mezi 1D a 2D variantou byla vytvořena varianta v 3D. Během vývoje bylo vytvořeno několik textur. Nakonec byla použita textura, díky níž figurky vypadají jako dřevěné.

Další vývoj by se mohl ubírat směrem urychlování metody. Metoda by se mohla paralelizovat. Také by se mohlo prozkoumat jak u různých primitiv použít metodu „Měkkého stínu jedním vzorkem“ a její použití u CSG operací. Tato metoda dosahuje velmi dobrých výsledků a její budoucí použití je velmi slibné.

Literatura

[1] GLASSNER, Andrew S.: *An introduction to ray tracing*. 1, London: Academic Press. Ltd., 1989. 327 p. ISBN 0-12-286160-4

[2] SHIRLEY, Peter; MORLEY, R. Keith: *Realistic ray tracing*. 2, USA: A.K.Peters. Ltd., 2003. 250p. ISBN 1-56881-198-5

[3] Photon mapping. Ve Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, 28. 1. 2003, poslední modifikace 22. 4. 2010 [cit. 2010-05-23]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Photon_mapping>

[4] STRACHOTA, Pavel. Saint-paul.fjfi.cvut.cz [online]. 2009 [cit. 2010-05-23]. Ray Tracing a další globální zobrazovací metody. Dostupné z WWW: <http://saint-paul.fjfi.cvut.cz/base/public-filesystem/admin-upload/POGR/POGR2/12.raytracing.pdf>

[5] Herakles.zcu.cz [online]. 2009 [cit. 2010-05-23]. Centre of Computer Graphics and Data Visualisation. Dostupné z WWW: <http://herakles.zcu.cz/education/zpg/cviceni.php?no=4>

[6] Phong reflection model. Ve Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, 9. 1. 2003 , poslední modifikace 7. 4. 2010 [cit. 2010-05-23]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Phong_reflection_model>

[7] Oren-Nayar Reflectance Model. Ve Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, 13. 1. 2007 , poslední modifikace 8. 5. 2010 [cit. 2010-05-23]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Oren%E2%80%93Nayar_Reflectance_Model>

[8] Blinn-Phong shading model. Ve Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, 5. 5. 2006 , poslední modifikace 15. 3. 2010 [cit. 2010-05-23]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Blinn%E2%80%93Phong_shading_model>

[9] NĚMEC, Martin. Barborka.vsb.cz [online]. [cit. 2010-05-23]. Osvětlení, stínování. Dostupné z WWW: <http://barborka.vsb.cz/nemec/zpg/prednasky/prednaska07.pps>

[10] Ocw.mit.edu [online]. 2004 [cit. 2010-05-23]. Transformation in Ray Tracing. Dostupné z WWW: <http://ocw.mit.edu/NR/rdonlyres/Electrical-Engineering-and-Computer-Science/6-837Fall2003/BD0F1D56-8FC9-4A06-A056-7596E4C79658/0/5_1_trans_hier.pdf>

[11] Constructive solid geometry. Ve Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, 3. 10. 2003 , poslední modifikace 17. 3. 2006 [cit. 2010-05-23]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Constructive_solid_geometry>

[12] Web.cs.wpi.edu [online]. 1997 [cit. 2010-05-23]. Distributed Ray Tracing. Dostupné z WWW: <http://web.cs.wpi.edu/~matt/courses/cs563/talks/dist_ray/dist.html>

[13] PARKER, Steven; SHIRLEY, Peter; SMITS, Brian. Cs.utah.ed [online]. 1998 [cit. 2010-05-23]. Single Sample Soft Shadows. Dostupné z WWW: <http://www.cs.utah.edu/~bes/papers/coneShadow/shadow.html>

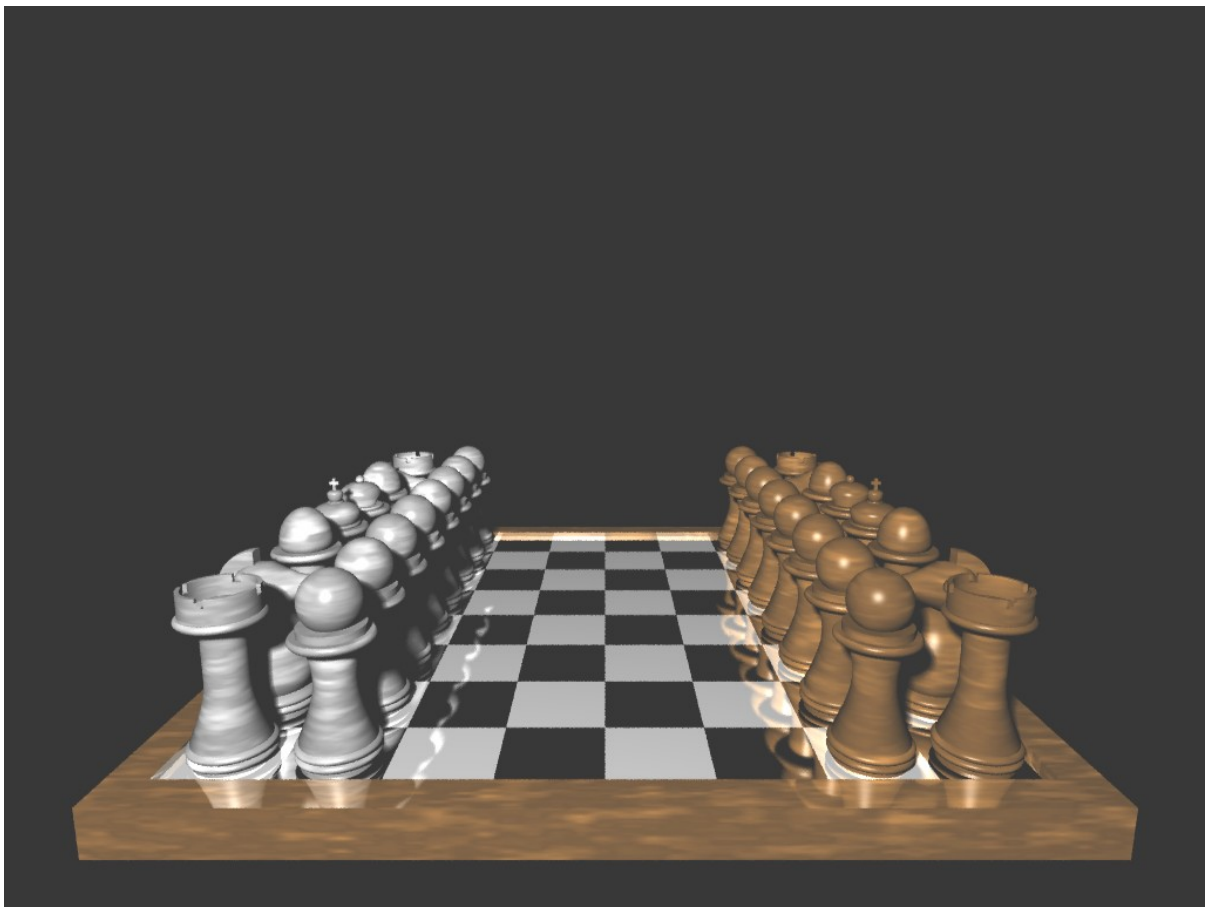
[14] Texturování. Ve Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, 13. 4. 2006, poslední modifikace 13. 5. 2010 [cit. 2010-05-23]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Texturov%C3%A1n%C3%AD>

[15] Procedural texture. Ve *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 23. 6. 2005, poslední modifikace 24. 4. 2010 [cit. 2010-05-23]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Procedural_texture>

[16] Freespace.virgin.net [online]. 1998 [cit. 2010-05-23]. Perlin Noise. Dostupné z WWW: <http://freespace.virgin.net/hugo.elias/models/m_perlin.htm>

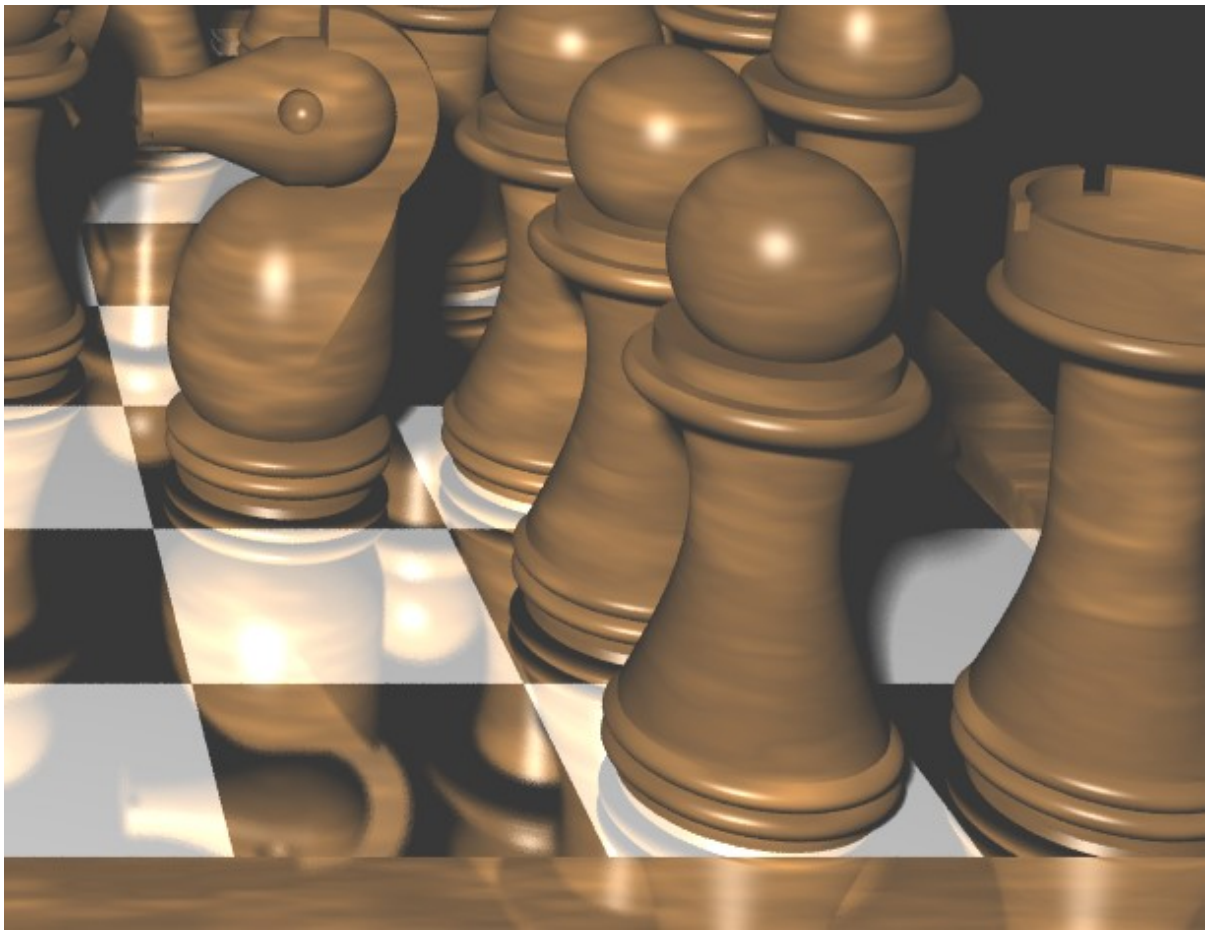
Seznam příloh

Příloha 1



Scéna 1

Scéna 2



Přiblížení ve scéně 2